Mobilité et bases de données

Etat de l'art et perspectives

Action Spécifique CNRS « Mobilité/Accès aux données »

G. Bernard, J. Ben-Othman, L. Bouganim, G. Canals, B. Defude, J. Ferrié, S. Gançarski, R. Guerraoui, P. Molli, P. Pucheral, C. Roncancio, P. Serrano-Alvarado, P. Valduriez

RÉSUMÉ.

Ce document est le résultat d'un travail collectif réalisé par les membres de l'action spécifique CNRS « Mobilité/Accès aux données ». Il fait le point sur l'état d'avancement des travaux réalisés dans le domaine des bases de données mobiles et identifie quelques orientations de recherche jugées particulièrement pertinentes. Compte tenu de la largeur du domaine, ce document ne prétend cependant pas à l'exhaustivité. Après une classification des applications mobiles et une caractérisation de leur environnement d'exécution, les problématiques suivantes sont analysées : gestion des données de localisation, modèles d'accès aux données, cohérence des traitements et synchronisation de copies, gestion de données embarquées, confidentialité des données et enfin adaptabilité des infrastructures logicielles à la mobilité et à la déconnexion.

MOTS-CLÉS : *Bases de données mobiles, gestion de la localisation, modèles d'accès aux données, cohérence et synchronisation, systèmes embarqués, confidentialité, adaptabilité.*

1. Introduction

La généralisation des télécommunications sans fil combinée à la prolifération des calculateurs ultra-légers connectables à l'Internet (assistants personnels, téléphones cellulaires, cartes à puce …) ou bien connectés en réseaux locaux (domotique, automobile, avionique …) amène une profonde mutation dans la conception, le déploiement et l'utilisation des systèmes d'information. A titre d'exemple, le rapport "Internet du futur" du RNRT estime qu'un citoyen des pays développés utilisera environ 80 processeurs quotidiennement par le biais de systèmes enfouis, souvent mobiles. Cette évolution technologique doit permettre à un utilisateur d'accéder à des données et d'exécuter des traitements n'importe où, n'importe quand et à partir de n'importe quel terminal.

Les applications de ce nouveau type de traitement, souvent appelé *ubiquitous computing* ou *pervasive computing*, sont multiples. Il peut s'agir d'applications personnelles dans lesquelles un utilisateur veut pouvoir accéder à tout moment à des données publiques (météo, trafic routier, cours de la bourse …) ou privées (données bancaires, agenda, dossier médical, bookmarks …). Il peut également s'agir d'applications professionnelles dans lesquelles des employés itinérants doivent accéder et partager en permanence, et où qu'ils se trouvent, des données relatives à leur entreprise ou à la tâche qu'ils exécutent en commun.

Ces besoins sont à mettre en opposition avec les contraintes fortes induites par l'environnement matériel et logiciel utilisé : faible débit des réseaux hertziens, déconnexions fréquentes (volontaires ou non), faibles capacités des terminaux mobiles en termes d'affichage, d'autonomie électrique, de puissance de traitement et de stockage, inadéquation des intergiciels conçus jusqu'à présent pour interconnecter des clients et des serveurs fixes.

La confrontation des besoins et des contraintes génère de multiples problèmes de recherche difficiles. Bien que la communauté bases de données s'intéresse au problème de la mobilité depuis une dizaine d'années, il apparaît de plus en plus clairement que nous n'en sommes qu'à la genèse de cette thématique et que le champ d'investigation reste immense. L'objectif de ce document est de faire le point sur l'état d'avancement des travaux réalisés dans le domaine des bases de données mobiles et d'identifier quelques orientations de recherche que nous avons jugées particulièrement pertinentes.

Ce document est structuré comme suit. La section 2 propose une classification des applications mobiles, introduit une architecture logicielle et matérielle de référence et caractérise les dimensions du problème de la mobilité dans les accès aux données. La section 3 adresse le problème de la gestion de données spatio-temporelles, utilisées pour la localisation d'entités mobiles. La section 4 porte sur les nouveaux modes d'accès à l'information permis, ou parfois imposés, par les environnements mobiles. Les sections 5 et 6 traitent du problème de cohérence des

traitements en environnement mobile ; la section 5 aborde ce problème du point de vue de la synchronisation de mises à jour effectuées par des mobiles en mode déconnecté alors que la section 6 aborde ce problème sous l'angle de transactions impliquant des mobiles au cours de leur exécution. La section 7 traite des problèmes liés à la gestion de données embarquées dans des calculateurs mobiles ultra-légers. La section 8 aborde les problèmes de confidentialité des données qui, bien que non directement liés à la mobilité, sont exacerbés dans ce contexte. La section 9 discute de la façon d'introduire un degré d'adaptabilité dans les applications et les ressources utilisées (intergiciels, réseaux, bases de données …) afin de supporter au mieux les conséquences de la mobilité. Enfin, la section 10 conclut cette étude.

2. Dimensions du problème

2.1. *Classification des applications mobiles*

Compte tenu de la variété des applications ayant trait à la mobilité, il est très difficile d'en faire une classification permettant de tirer des conclusions précises sur chaque classe. La classification qui nous a paru la plus pertinente utilise comme critère discriminant le fait que la mobilité soit liée à l'utilisateur, au serveur de données ou aux données elles-mêmes.

- *Client mobile – serveur fixe :* cette classe couvre en premier lieu les applications client-serveur traditionnelles dans lesquels certains utilisateurs sont mobiles. Il peut s'agir d'applications professionnelles dans lesquelles des employés itinérants doivent rester en contact avec le système d'information de l'entreprise (ex : voyageur de commerce, expert d'assurance …). Il peut également s'agir d'applications personnelles dans lesquelles un utilisateur fait appel aux services d'un hébergeur de données ou *DSP* (*Database Service Provider*) qui va lui garantir un stockage sûr et un accès permanent à ses données privées (données bancaires, agenda, dossier médical, bookmarks …). Enfin, rentrent dans cette classe des applications effectuant de la dissémination d'information à destination d'un large public d'utilisateurs mobiles (diffusion d'informations météo, trafic routier, cours de la bourse …). Même si certaines données sont temporairement répliquées sur des mobiles afin d'autoriser des traitements en mode déconnecté, la copie primaire des données est gérée par le serveur fixe dans tous les cas.
- *Client mobile – serveur mobile :* cette classe couvre les applications dans lesquelles client, serveur et données sont tous embarqués dans un calculateur mobile. Les dossiers portables de différentes natures sont des exemples typiques de cette classe d'application. Selon le terminal employé, ces dossiers portables peuvent devenir fortement sécurisés (ex : dossier médical sur carte à puce ou carnet d'adresses dans une carte SIM). La copie primaire des données est ici gérée par le serveur mobile même si certaines données peuvent être répliquées sur le réseau fixe pour des raisons de tolérance aux défaillances. Bien que les applications visées soient majoritairement personnelles, et donc mono-

utilisateur, on peut envisager des traitements coopératifs entre serveurs mobiles (ex : prise de rendez-vous sur un ensemble d'agendas embarqués dans des assistants personnels).

- *Client fixe – serveur mobile :* on peut ranger dans cette classe des applications où des capteurs attachés à des mobiles diffusent des informations sur leur état à destination d'une unité de traitement fixe et reçoivent éventuellement des commandes émanant de cette unité (ex : contrôle à distance d'un véhicule). On peut également imaginer des applications statistiques réalisant de l'extraction de données détenues par plusieurs serveurs mobiles (ex : étude épidémiologique sur une large population de dossiers médicaux embarqués sur des cartes à puce).
- *Données mobiles dans un serveur fixe :* par abus de langage, nous utilisons ici le terme données mobiles pour caractériser des applications dans lesquelles les données gérées correspondent à la localisation de mobiles. De telles bases de données de localisation peuvent être exploitées dans des domaines aussi variés que la localisation de téléphones cellulaires par les opérateurs télécoms, la gestion de flottes de véhicules (taxis, camions de livraison) ou le suivi de la migration d'animaux bagués.

2.2. *Environnement logiciel et matériel de référence*

Cette section introduit un environnement typique permettant de réaliser des accès itinérants à des données. Bien que non exclusif, cet environnement sera utilisé comme référence dans la suite du document.

Du point de vue matériel, les équipements suivants sont considérés :

- *Terminaux :* il s'agit de calculateurs légers et mobiles (ordinateur portable, assistant personnel, téléphone cellulaire, carte à puce, ordinateur de bord …) dotés d'une interface radio leur permettant de se connecter au réseau filaire. Bien qu'à une échelle différente, les caractéristiques de ces terminaux sont : capacités de calcul et de stockage réduites, faible autonomie électrique, capacité d'affichage réduite, architecture matérielle ad-hoc rendant certains traitements très coûteux (ex : écritures dans des mémoires stables de type EEPROM ou Flash), faible fiabilité (risques de perte, vol, destruction).
- *Serveurs :* il s'agit de calculateurs traditionnels, connectés de façon permanente au réseau filaire et dotés de capacités de traitement et de stockage importantes.
- *Réseau hertzien :* on considère ici un réseau de communication cellulaire dont les caractéristiques sont : faible débit (de quelques Kbps à quelques Mbps suivant la génération de réseau), qualité de transmission médiocre (taux d'erreurs élevé), déconnexions accidentelles (passage dans une zone d'ombre). Quand le taux de déconnexions accidentelles est élevé, on parle de *connexion faible* au réseau filaire. Par contre, ce médium est bien adapté à la diffusion de messages.
- *Réseau filaire :* par opposition au réseau hertzien, le réseau filaire est supposé offrir un débit et une qualité de communication élevés.

Du point de vue logiciel, on distingue les entités suivantes :

- *Clients :* on qualifiera de client un logiciel applicatif interagissant directement, ou indirectement par le biais d'un médiateur, avec des sources de données fixes ou mobiles.
- *Sources de données :* on regroupe sous cette terminologie le processus serveur et la source de données qu'il gère. Une source de données est capable d'émettre des données en réponse à une requête émanant d'un client ou d'un médiateur.
- *Médiateur :* on regroupe sous ce terme un ensemble d'intergiciels permettant à un client de s'interfacer avec une ou plusieurs sources de données. Le médiateur peut ainsi jouer des rôles variés parmi lesquels : intégrer des sources de données hétérogènes pour en offrir une image uniforme, virtuelle (schéma de médiation) ou concrète (entrepôt de données) ; exécuter des requêtes complexes sur cette image ; notifier les clients lorsqu'un événement satisfait une de leur souscription ; organiser la dissémination de l'information ; assurer des tâches d'administration de ressources (adaptation des flux à destination des clients, découverte de services, rafraîchissement des caches clients …).
- *Proxy client :* il s'agit d'un agent représentant un client mobile sur le réseau filaire. Cette représentation peut prendre plusieurs formes, comme par exemple le masquage de la déconnexion d'un client en enregistrant les messages qui lui sont destinés jusqu'à la prochaine connexion.
- *Proxy médiateur :* il s'agit d'un agent représentant le médiateur sur le terminal d'un client mobile. Cette représentation a pour objectif de permettre au client d'exécuter des traitements en mode déconnecté, en masquant autant que faire se peut la déconnexion.

2.3. *Dimensions du problème*

En rapprochant les besoins introduits par les classes d'applications présentées en section 2.1 des contraintes liées à l'architecture logicielle et matérielle présentées en section 2.2, on peut identifier les dimensions suivantes du problème de la mobilité dans les accès aux données :

- *Gestion de données spatio-temporelles :* un des corollaires de la mobilité est la localisation des équipements mobiles (téléphones cellulaires, flottes de véhicules…). Les bases de localisation sont des bases de données particulières dont l'objectif est de retrouver rapidement la localisation d'un mobile ou du moins de l'approximer avec une faible marge d'erreur. Alors que le déplacement d'un mobile est généralement un événement continu, son reflet dans la base de données ne peut être que discret pour d'évidentes raisons de performance. Cette approximation a un impact fort sur la modélisation des données (i.e., de leur mouvement), sur l'expression des requêtes (ex : trouver les hôtels les plus proches de la position que l'utilisateur aura atteinte dans une heure) et sur leur évaluation efficace nécessitant des opérateurs et des méthodes d'indexation ad-hoc.

- *Modèles d'accès à l'information :* Si le modèle client/serveur traditionnel est toujours possible en environnement mobile, d'autres modèles d'accès aux données sont envisageables. Par exemple, le modèle *publish/subscribe* permet à un client d'être notifié automatiquement lorsqu'un événement satisfaisant un de ses abonnements est publié (émis) sur le réseau filaire. On peut également envisager qu'un serveur diffuse de façon répétitive une information et que les clients intéressés se mettent à son écoute. Ces modèles offrent des avantages du point de vue de la consommation de bande passante et du support de l'asynchronisme lié aux déconnexions, au détriment de la latence. Enfin, des modèles basés sur une diffusion épidémique de l'information peuvent être envisagés pour supporter le passage à l'échelle (très grand nombre de clients) tout en augmentant la disponibilité des données (éviter les points uniques de défaillance).
- *Synchronisation et traitements déconnectés :* la mobilité peut être caractérisée par une alternance de phases de travail en mode connecté et en mode déconnecté. Des mises à jour peuvent ainsi intervenir sur des copies d'objets embarqués sur un calculateur mobile déconnecté. Il s'agit donc d'un modèle de réplication symétrique dans lequel les différentes copies ne sont pas accessibles de façon synchrone. La divergence de copie devient alors inévitable. Lors des reconnections, des protocoles sophistiqués de synchronisation/réconciliation doivent être appliqués pour rétablir la convergence des copies. La qualité de ces protocoles se mesure par leur finesse et leur degré d'automatisation.
- *Cohérence transactionnelle :* Les protocoles de synchronisation garantissent un degré de cohérence des données basé sur la propriété de convergence des copies. Les plus sophistiqués d'entre eux respectent également les règles de causalité (les opérations liées causalement sont exécutées dans le bon ordre) et de préservation de l'intention de l'utilisateur (sous réserve que celle-ci soit caractérisable). Cependant, certaines applications ont des besoins plus fort de cohérence qui nécessitent le respect des propriétés transactionnelles ACID (Atomicité, Cohérence, Isolation, Durabilité). La mobilité a cependant un impact majeur sur la mise en œuvre de chacune de ces propriétés.
- *Gestion des données embarquées :* Concevoir des composants bases de données destinés à être embarqués sur des calculateurs mobiles ultra-légers est un challenge important. En effet, chaque calculateur est architecturé de sorte à satisfaire des propriétés précises (portabilité, autonomie électrique, sécurité, adaptation à des traitements spécifiques…) en tenant compte de contraintes matérielles imposées (taille maximale de la puce, débit de communication contraint, technologie mémoire imposée…). Par ailleurs, ces architectures sont en permanente évolution pour couvrir les besoins de nouvelles applications. Il est donc tout aussi important de définir des moteurs bases de données adaptés à ces architectures spécialisées que d'exprimer des recommandations pour les concepteurs de ces architectures.

- *Confidentialité des données :* lorsque les données sont trop volumineuses pour être embarquées ou qu'elles sont partagées entre plusieurs utilisateurs, le moyen le plus simple pour les rendre accessibles de tout endroit est de les stocker sur un serveur fixe, par exemple géré par un DSP (Database Service Provider) et d'y accéder via l'Internet. Se pose alors le problème de la confidentialité de ces données. Des modèles de sécurité de bout en bout (du mobile à la donnée stockée) restent à définir afin de résister à tous types d'attaques, y compris celles dirigées vers l'empreinte disque de la base de données, qu'elles soient menées par un intrus ou par un administrateur de données. Ces modèles de sécurité imposent une parfaite coordination entre les protocoles de transport réseaux, les outils de médiation et les sources de données.
- *Adaptabilité :* l'objectif de l'adaptabilité est d'offrir à l'utilisateur, quelle que soit sa localisation et son type de terminal, un service aussi proche que possible de celui fourni dans un contexte statique, où le terminal de l'utilisateur est fixe, dispose de ressources importantes et est relié aux serveurs par un réseau fiable et performant. L'adaptabilité a un impact à tous les niveaux : application, intergiciels, couche réseau et sources de données. Par exemple, la précision d'un objet téléchargé sur un terminal peut dépendre des capacités d'affichage et de stockage du terminal utilisé ainsi que du temps de connexion nécessaire au téléchargement. Le masquage de la déconnexion à l'utilisateur est également un soucis majeur. Enfin, l'objectif étant de fournir à l'application le meilleur mode de fonctionnement en fonction de l'état courant des ressources, il faut être capable d'acquérir dynamiquement de l'information sur cet état.

  Chacune de ces dimensions fait l'objet d'une section dans la suite du document.

3. Données spatio-temporelles

3.1. *Motivations*

Dans un contexte d'applications mobiles, on peut avoir à prendre en compte non seulement la mobilité des clients mais aussi la mobilité des données que vont interroger les clients. Par exemple, dans une application de gestion de flottes de camions de transport, on va vouloir poser des requêtes du type "retrouver tous les camions qui sont actuellement à moins de 20 kilomètres de tel entrepôt". Dans le cadre des nouveaux services proposés sur les téléphones mobiles, on trouve des exemples du type "visite interactive d'une ville" qui permet d'avoir sur son poste mobile un descriptif du monument devant lequel on se trouve. Ce qui change dans ces exemples par rapport aux applications classiques de bases de données, c'est que l'on doit prendre en compte un déplacement (déplacement de l'objet interrogé dans le premier exemple et déplacement du client dans le second) et/ou une dimension temporelle. De plus le déplacement est continu et ne peut être connu de manière sûre à l'avance. On peut classifier les requêtes impliquant de la mobilité en :

- requêtes où le client est mobile et les données sont fixes (ex : informations sur l'endroit où se situe l'utilisateur ; liste des restaurants dans un rayon de 500 mètres …) ;
- requêtes où les données sont mobiles et le client est fixe (ex : liste des camions qui vont arriver dans telle zone dans 10 minutes) ;
- requêtes où les données et le client sont mobiles (ex : liste des 10 taxis les plus proches de l'endroit où l'utilisateur sera dans 5 minutes).

3.2. *Problématique*

La problématique de recherche est voisine de celle des bases de données spatio-temporelles puisque l'on doit manipuler ces deux dimensions à la fois. La caractéristique particulière ici est que l'on a affaire à des données pouvant se déplacer de manière continue. Les problèmes posés sont les suivants [Wolfson et al, 1998] :

- modélisation de la localisation (et donc du mouvement) : la valeur de la position des données ne peut être maintenue de manière continue sans compromettre grandement les performances du système (le volume de données va vite devenir prohibitif et la bande passante requise pour transporter les nouvelles positions va elle aussi être importante). Il faut donc pouvoir calculer les nouvelles positions via des fonctions d'approximation prenant en compte le temps ;
- puissance d'expression du langage de requêtes : les requêtes nécessitent de manipuler à la fois la dimension spatiale (points, lignes, régions, polygones) et le temps ;
- indexation des données : des index ad-hoc sont nécessaires pour une évaluation efficace des requêtes spatio-temporelles. Les index classiques ne répondent pas au problème car les données évoluent continuellement, rendant impossible la mise à jour des index avec la même fréquence ;
- incertitude/imprécision : par définition il est impossible d'avoir des certitudes sur la position future d'un objet et il est donc nécessaire d'évaluer la précision du positionnement calculé. Ceci peut être pris en compte au niveau du langage de requêtes (introduction de modalités comme sûrement, possiblement, probablement, …) ou au niveau des performances du système (fixer l'équilibre entre la fréquence des mises à jour et le niveau d'imprécision induit).

3.3. *Solutions proposées*

La représentation des données spatio-temporelles suit deux courants principaux. Le premier est basé sur les bases de données contraintes et consiste en un modèle de données abstrait non spécialisé et un langage de requêtes [Grumbach et al, 1998]. Le modèle est basé sur des contraintes linéaires et permet la représentation de données avec un nombre arbitraire de dimensions. Les opérations sont celles de l'algèbre relationnelle étendue à un nombre infini de points. Le second courant suit une

approche type abstrait de données avec des types de base et des types spatiaux ainsi que des versions temporelles de ceux-ci [Guting et al, 2000]. L'intérêt du modèle proposé est qu'il permet de modéliser aussi bien des points que des régions mobiles. [Wolfson et al, 1998] définit un modèle de données appelé MOST permettant de modéliser la mobilité sous forme d'attributs dynamiques, c'est-à-dire d'attributs dont la valeur change continuellement dans le temps sans être explicitement modifiée. Ils sont représentés par des fonctions du temps. Les requêtes sont exprimées à l'aide du langage FTL qui permet d'interroger des états futurs de la base de données. Une requête FTL peut être soumise selon différents modes (instantané, continu, persistant) qui vont donner des résultats différents pour cette même requête.

Plusieurs index ont été proposés pour les données mobiles. La plupart modélisent la trajectoire des objets mobiles sous forme de rectangles englobant paramétrés par le temps. Un problème crucial concerne la suppression de valeurs dans cet index. [Saltenis et al, 2002] propose d'associer une date d'expiration à chaque rectangle qui est ensuite utilisée pour faire des suppressions. Ils utilisent comme structure de données une variante des arbres R^* appelée R^{EXP}.

La localisation des données entre deux observations est généralement calculée par une fonction d'interpolation. Le biais introduit par cette fonction ainsi que par les instruments de mesure peut être estimé [Pfoser et al, 1999].

3.4. *Perspectives*

La modélisation de données mobiles touche de nombreux domaines d'applications (systèmes d'informations géographiques, applications multimédia de type vidéo, vision pour la robotique, …). Chaque domaine a développé ses propres outils, mais il reste maintenant à rassembler ces approches diverses dans un cadre unifié. Les modèles proposés restent à un niveau d'abstraction rendant difficile des implantations dans le cadre de SGBD et il reste du travail à faire dans ce sens.

Les techniques d'indexation sont encore récentes et nécessitent d'être validées de manière expérimentale, ce qui suppose le développement de bancs d'essais pour ce domaine.

Enfin, la puissance d'expression des langages de requêtes proposés n'est pas encore formellement étudiée. De même, l'intégration des constructions proposées dans un langage destiné aux utilisateurs (que ce soit sous forme d'un SQL étendu ou d'un autre langage) n'est pas faite. Ces langages peuvent être étendus pour essayer de prendre en compte l'incertitude des données manipulées [Abdessalem et al, 2001].

4. Modèles d'accès à l'information

4.1. *Problématique*

Le cœur d'un système d'information a traditionnellement été basé sur un serveur de bases de données centralisé. Des processus clients accèdent à ce serveur pour stocker et retrouver l'information. Ce type d'interaction client-serveur est typiquement effectué de manière synchrone : lorsqu'un client invoque une requête sur le serveur de base de données, il arrête ses activités en attendant le résultat de sa requête. Parfois, une variante asynchrone est adoptée. Cette dernière consiste à ce que le client souscrive au serveur de base de données en exprimant le type d'information auquel il est intéressé. A chaque fois qu'une information de ce type est disponible, le client en question en est informé. On parle aussi dans ce cas de schéma de communication par publication-souscription et de base de données active.

Dans un environnement mobile et dynamique, une solution synchrone ou asynchrone, basée sur serveur central de base de données n'est pas toujours optimale. D'une part, ce serveur peut rapidement constituer un goulot d'étranglement et rendre extrêmement lent l'accès à l'information. D'autre part, ce serveur constitue un point central de défaillance. Même si ce serveur est mis en œuvre à travers un cluster de machines performant et tolérant aux défaillances, il peut tout simplement devenir inaccessible à cause d'une défaillance du réseau. On peut envisager une solution répartie dans laquelle le serveur est dupliqué sur des machines physiquement distantes et proches, du point de vue de l'accessibilité du réseau, des sites potentiels des clients mobiles. Cette solution permet de contourner un peu le problème d'accessibilité aux données mais reste limitée car les clients sont mobiles et il est difficile de déterminer les sites adéquats à la duplication du serveur central.

4.2. *Solutions proposées*

Une solution plus ambitieuse consiste à faire en sorte que le serveur de base de données soit mis en œuvre par tous les processus participant à l'application : la base de données est partout, y compris sur les clients. Même si certains processus continuent à jouer un rôle unique (client ou serveur), l'idée fondamentale est précisément d'éliminer la dichotomie entre clients et serveurs. On ne parle plus que de processus.

La diffusion et la recherche d'information se font alors de manière symétrique : chaque processus, de manière périodique, contacte d'autres processus et leur communique une partie de l'information dont il dispose. Pour passer à l'échelle, on fait en sorte qu'un processus ne diffuse pas de l'information à tous les processus avec lesquels il peut communiquer, mais à un sous-ensemble choisi au hasard. La dissémination de l'information entre des processus ressemble à la dissémination

d'un virus entre individus et peut être étudiée de manière similairement probabiliste [Bailey, 1975].

Pour rendre le sujet plus concret, considérons une application financière permettant à des boursicoteurs amateurs, mobiles et disséminés à travers le monde, d'échanger de l'information sur des actions de la bourse et tout ce qui a trait aux entreprises associées à ces actions. L'information essentielle est produite à New York, Londres, Tokyo et Paris, mais aussi par les participants eux-mêmes qui peuvent échanger leurs expériences et leurs analyses. Bien entendu, une difficulté majeure est d'assurer que l'information est disséminée de manière efficace pour atteindre le maximum de participants intéressés. A large échelle et dans un environnement mobile, il est impossible d'assurer qu'aucun participant ne rate jamais l'information à laquelle il est intéressé. Cela demanderait des mécanismes de diffusion qu'il est impossible de mettre en œuvre avec des performances acceptables. Un algorithme de dissémination épidémique permet justement d'allier des performances acceptables et une probabilité d'accès à l'information très raisonnable.

L'histoire a démontré que les épidémies sont des fléaux redoutables. Il suffit qu'un petit nombre d'individus soient contaminés et se déplacent pour que le fléau se propage à très grande vitesse. Les épidémies sont aussi très résistantes aux défaillances lors du processus d'infection. De nombreux travaux ont été consacrés à l'analyse, l'observation et la conception de modèles mathématiques épidémiques. Appliqué à la diffusion d'information, un tel processus épidémique se rapproche d'un modèle de diffusion de rumeurs [Pittel, 1987], parfois appelé téléphone Arabe.

Appliquer une méthode de dissémination épidémique de l'information dans un système mobile et dynamique est très séduisant : l'idée a été initialement proposée au laboratoire Xerox de Palo Alto [Demers et al, 1987] puis a été relancée ces dernières années à l'Université de Cornell [Birman et al, 1999]. Chaque processus stocke l'information reçue dans un tampon d'une capacité donnée (k1) et retransmet cette information un nombre limité de fois (k2). A chaque fois, le processus en question transmet l'information à un ensemble de processus de taille (k3) choisi au hasard. Etant donné que k1, k2 et k3 sont fixés et ne dépendent pas de la taille du système (e.g., du nombre de processus total), le passage à l'échelle de tels algorithmes est évident. De plus, aucun processus ne constitue un goulot d'étranglement ou un point central de défaillance. La source de la fiabilité est le choix arbitraire du sous-ensemble de processus auquel l'information est disséminée.

Même si l'idée est très intuitive, son application dans un environnement informatique est loin d'être triviale. De nombreuses questions se posent : (1) comment les processus viennent-ils à se connaître pour pouvoir se transmettre l'information ? (2) quelle information un processus doit-il éliminer lorsque son tampon est plein ? (3) comment prendre en compte l'intérêt des processus et diminuer ainsi la probabilité qu'un processus non intéressé par une information doive néanmoins la gérer ? Nous discutons dans ce qui suit de certaines réponses à ces questions en ouvrant quelques perspectives de recherche. Ces perspectives ne

sont pas indépendantes mais nous les présentons comme telles pour faciliter leur compréhension.

4.3. *Perspectives*

Connaissance des processus

Dans l'algorithme de dissémination proposé à l'Université de Cornell, [Ozkasap et al, 1999], on suppose que chaque processus connaît au préalable tous les autres. Ceci est réaliste dans un cluster de processus mais pas dans un environnement mobile. Une manière plus adaptée pour gérer ce problème consiste à mixer la dissémination d'information et la connaissance des autres processus [Eugster et al, 2001]. On suppose pour cela que chaque processus en connaît certains autres. Lorsqu'un processus transmet une information à un autre processus, il transmet aussi une liste de processus qu'il connaît. Cette méthode pose néanmoins au moins trois problèmes.

- Comment un processus connaît-il initialement d'autres processus ? Ceci requiert un mécanisme externe, de type pages jaunes, dont la prise en compte dans la modélisation probabiliste de la dissémination est délicate.
- Dans les algorithmes de dissémination épidémique, on suppose en général que le fait que deux processus se connaissent suffise pour qu'ils puissent communiquer. Dans un environnement mobile, il serait plus réaliste d'introduire une notion de connaissance basée sur la possibilité de communiquer.
- La fiabilité de la dissémination épidémique repose sur l'idée que chaque processus choisit au hasard un sous-ensemble de processus auquel il transmet son information. En supposant que chaque processus possède une probabilité égale d'être connu dans le système (hypothèse d'uniformité), on peut effectivement montrer qu'une fiabilité acceptable peut être atteinte. Néanmoins, assurer l'hypothèse d'uniformité dans un environnement mobile et dynamique est une question ouverte.

Le problème du stockage

L'idée de base des algorithmes épidémiques consiste pour chaque processus à stocker tous les messages qu'il obtient et à les retransmettre un nombre limité de fois. La capacité de stockage étant limitée, et pour assurer le passage à l'échelle de l'algorithme, chaque processus doit pouvoir éliminer des messages. Chaque processus doit ainsi garder de manière permanente k1 messages au maximum, doit retransmettre chacun k2 fois au maximum, et à chaque fois à un sous-ensemble de processus de taille k3. En fonction du taux de diffusion des nouveaux messages, la capacité de stockage de chaque processus (k1) peut s'avérer insuffisante pour assurer que chaque message est diffusé un nombre suffisant de fois pour garantir sa transmission fiable. Soit l'on considère une stratégie conservatrice dans laquelle les nouveaux messages sont éliminés si la limite de stockage est limitée, soit l'on

considère une stratégie plus dynamique selon laquelle les messages les plus vieux sont éliminés.

- Une idée prometteuse, expérimentée dans [Kouznetsov et al, 2001], est d'associer la notion d'âge d'un message au nombre de fois où ce message a été transmis. Une approche complémentaire consiste à utiliser la sémantique des messages afin de faciliter le processus de décision du message à détruire. Par exemple, un message peut en annuler un autre si son contenu rend le message précédent obsolète (e.g., une information plus fraîche). Enfin une troisième approche consiste à adapter dynamiquement le flux de diffusion de messages au taux de fiabilité atteint par l'algorithme de diffusion. Le problème délicat est de faire en sorte que le retour à l'application reflète bien la fiabilité du mécanisme, sans pour autant introduire de mécanisme de communication explicite entre l'application et le mécanisme de diffusion.

Le problème du filtrage

Jusqu'à maintenant, nous avons considéré que le but de la diffusion épidémique était d'assurer que chaque message atteigne le plus grand nombre de processus. Ce principe a un sens si tous les processus expriment un intérêt similaire pour les informations stockées. Si par contre les processus expriment une préférence entre les messages, il faut à la fois maximiser la probabilité p1 qu'un processus obtienne tous les messages qui l'intéressent et minimiser la probabilité p2 qu'il reçoive les messages qui ne l'intéressent pas [Eugster et al, 2002]. Ceci nécessite d'introduire des mécanismes de filtrage au sein du processus de diffusion épidémique. Ces derniers devront ainsi être utilisés en même temps que les mécanismes de choix probabiliste. Une telle combinaison pose au moins deux problèmes difficiles : (1) comment un processus peut-il savoir quel message intéresse un autre processus ? (2) ne pas diffuser à un processus *P* un message *m* qui ne l'intéresse pas afin de minimiser la probabilité p2 à un impact sur la probabilité p1 si *P* connaît un processus intéressé par ce message et que l'émetteur du message m ne connaît pas.

5. Synchronisation et traitements déconnectés

5.1. *Motivations*

La mobilité peut être caractérisée par une alternance de phases de travail en mode connecté et en mode déconnecté. Une façon de permettre le travail en mode déconnecté consiste à répliquer les objets sur le support mobile avant la déconnexion et à réconcilier les copies de ces objets à la reconnexion. Nous avons identifié deux exemples typiques de cette façon de travailler :

- *le travail individuel multi-postes :* du fait de la multiplicité des calculateurs portables (PDA, Laptop, téléphones cellulaires …), un usager est de plus en plus amené à manipuler des copies de ses propres objets (agenda, fichier) sur des

postes différents et à des instants différents, générant une divergence de ces copies.

- *le travail collaboratif multi-synchrone* : le travail collaboratif (appelé CSCW ou *groupware* [Ellis et al, 1989]) permet à un groupe d'usagers de manipuler et de modifier de façon cohérente des objets partagés (documents, plans). Ce mode de travail a surtout été étudié et pratiqué dans le cadre d'environnements temps réel dits synchrones dans lesquels chaque usager voit immédiatement les effets de ses propres actions sur sa copie de l'objet et aussitôt que possible les effets résultants des actions des autres. La mobilité favorise le développement d'un nouveau mode de travail collaboratif dit multi-synchrone [Dourish, 1995]. Ce mode est caractérisé pour l'usager par des périodes de déconnexion, pendant lesquelles il manipule seul sa copie, et par des périodes de connexion pendant lesquelles il diffuse aux autres ses mises à jour et où il intègre celles des autres.

Dans un cas comme dans l'autre, la déconnexion entraîne la divergence des copies. Le défi consiste donc à fournir des outils et à définir des méthodes permettant de réconcilier ces copies de manière la plus harmonieuse possible.

5.2. *Problématique*

Maintenir des copies d'un même objet sur plusieurs ordinateurs (i.e. sites) est une technique fréquemment utilisée pour accroître la disponibilité et les performances [Gray et al, 1996]. On considère généralement que tous les sites sont connectés en permanence, de sorte que la cohérence des copies d'un même objet est assurée :

- soit par des stratégies dites *pessimistes*, qui répercutent immédiatement et en exclusion mutuelle, chaque mise à jour sur toutes les copies et qui bloquent les requêtes de lecture durant toute la durée de la modification,
- soit par la diffusion des mises à jour à tous les sites et leur intégration dans le même ordre sur chaque site (on parle alors de *diffusion atomique*).

La mobilité rend plus difficile le maintien de la cohérence des copies dans la mesure où tous les sites ne sont pas connectés simultanément, les connexions sont temporaires, les déconnexions souvent longues et parfois involontaires. Elle nécessite des approches dites *optimistes* [Saito et al, 2002], caractérisées par l'existence de copies divergentes. Pour converger, ces approches utilisent la diffusion en différé des mises à jour (par exemple par des algorithmes dits *épidémiques*, cf. section 3) et leur intégration sur les copies dans des ordres différents. Lorsque les mises à jour sont conflictuelles, interdisant la convergence automatique des copies, il faut avoir recours à des phases de réconciliation manuelle.

5.3. *Solutions proposées*

Synchroniseurs de fichiers

Lorsqu'il utilise un Synchroniseur [Balasubramaniam et al, 1998], l'usager est conscient que les copies de son objet ne sont pas nécessairement dans le même état. Le Synchroniseur lui permet de contrôler explicitement la réplication de son objet en lui fournissant un outil pour détecter les conflits, propager les mises à jour non conflictuelles entre ces copies et assurer la réconciliation dans le cas contraire. Les Synchroniseurs de fichiers tels que Microsoft's Briefcase, Power Merge, Windows File Synchronizer ou encore Unisson considèrent une hiérarchie de fichiers. Ils permettent de propager d'une copie à l'autre, les créations /destructions de fichiers ou de catalogues et leurs modifications, sauf quand elles portent sur deux copies d'un même fichier.

Synchroniseurs de données

Les Synchroniseurs de données (HotSync de Palm Pilot, Intellisync de Puma Technology), plus spécifiques, permettent d'aller plus loin et de fusionner les mises à jour portant sur les copies d'un même fichier mais nécessitent une connaissance précise de son type et de sa structure (i.e. agenda, carnet d'adresse, boîte à lettres,...). Dans ce contexte, la spécification SyncML d'un protocole de synchronisation portable sur n'importe quel support ne change pas fondamentalement le problème.

Bien qu'ayant le mérite d'exister, les Synchroniseurs (de fichiers ou de données) souffrent de nombreux inconvénients : ils ne peuvent réaliser la fusion que de deux copies à la fois, ils sont spécifiques à des systèmes et à des types d'objets bien particuliers, leur détection des conflits est très primitive et enfin ils ne prennent en compte que l'état de l'objet et sa modification, sans considérer les opérations faites sur l'objet et leur sémantique (augmentant par là le nombre des situations de réconciliation).

Gestionnaires de configuration

Les gestionnaires de configuration [Estublier, 1995] permettent à deux utilisateurs d'avoir des copies divergentes d'un même objet par le paradigme "copier-modifier-fusionner". Un utilisateur commence par copier une version d'un objet multi-versionné dans son espace de travail. Il peut alors modifier sa copie avec ses outils habituels. Avant de rendre publiques ses modifications, il doit réconcilier l'état de sa copie avec la version courante présente dans le gestionnaire de configuration. La phase de fusion est alors obtenue par l'exécution d'un outil de fusion au sein de l'espace de travail. Un outil de fusion a en paramètre trois versions du même objet : la version locale, la version courante présente dans le gestionnaire et enfin la version correspondant au premier ancêtre commun à la version locale et à la version courante. La phase de fusion proprement dite est ici réalisée par des outils dédiés.

Outils de fusion

Plusieurs outils de fusion [Munson et al, 1994] sont disponibles actuellement pour réaliser la fusion de fichiers texte (diff3, rcsmerge [Tichy, 1985], FileResolve [Feiler et al, 1990]), de fichiers XML (xmlmerge [XML, 2000]) ou de fichiers UML (Rational Rose). Ces outils peuvent être considérés comme des Synchroniseurs de données et sont actuellement utilisés dans le domaine du travail coopératif et dans celui du génie logiciel.

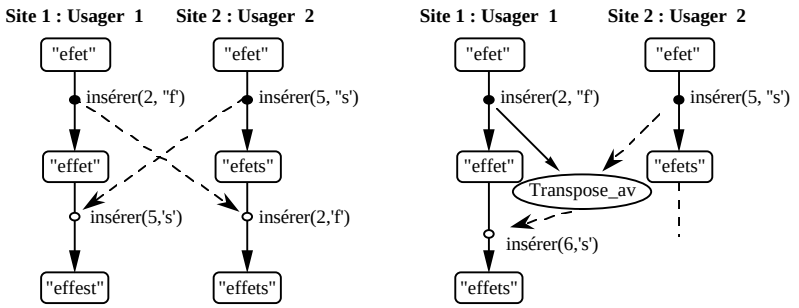
Algorithmes collaboratifs synchrones

Pour garantir la cohérence des copies d'un objet dans les environnements répartis collaboratifs synchrones, les méthodes mises en jeu [Sun et al, 1998], [Vidot et al, 2000], exploitent les opérations faites sur l'objet typé et leur sémantique, et les mémorisent dans des histoires. Elles satisfont en outre trois propriétés : (1) respect de la causalité, (2) respect de l'intention et (3) convergence des copies.

Dans ces environnements, toute opération générée sur une copie est exécutée immédiatement, avant d'être diffusée, reçue et exécutée sur les autres copies. Nous présentons brièvement les problèmes posés par ces propriétés et les techniques utilisées pour les résoudre sur un exemple d'édition de texte collaborative.

Le *respect de la causalité* garantit que les opérations entre lesquelles existe une relation de précédence causale, sont sérialisées dans le même ordre sur chaque copie. On dit que l'opération op_1 *précède causalement* l'opération op_2 ssi op_2 a été générée et exécutée sur un site après l'exécution de op_1 sur ce même site (autrement dit, l'effet de op2 dépend du résultat de op1). Si ces opérations sont délivrées et exécutées dans un ordre différent de l'ordre causal sur un autre site, alors leurs effets peuvent produire des résultats incohérents. Le respect de la causalité est résolu dans la plupart des méthodes par l'utilisation d'un vecteur d'état V_S associé à chaque site S et à chaque objet. Certaines méthodes utilisent des estampilles continues délivrées par un séquenceur, ce qui permet d'assurer, associé à une diffusion différée, une réception séquentielle compatible avec la réception causale.

Des opérations qui ne sont pas causalement reliées sont concurrentes. Elles peuvent de ce fait être exécutées sur les différentes copies dans des ordres quelconques. L'exécution locale d'opérations générées à distance doit cependant respecter l'*intention* de l'usager distant. Par exemple, considérons deux usagers travaillant simultanément sur des copies du même objet, dans le même état initial "efet". L'intention de l'usager 1 est d'ajouter un 'f' en position 2 pour obtenir "effet" ; son intention est réalisée par l'opération insérer(2,'f'). L'intention de l'usager 2 est d'ajouter 's' à la fin du mot ; elle est réalisée sur sa copie O_2 par l'opération insérer(5,'s'). Quand cette opération est délivrée et exécutée sur le site 1, le nouvel état résultant "effest" de la copie O_1 ne traduit pas l'intention de l'usager 2 (Fig. 1.a). Afin de respecter son intention, il est nécessaire de transformer cette opération sur le site 1 et d'exécuter insérer(6,'s') au lieu de insérer(5,'s').



a) Non respect de l'intention de l'usager 2, sur le site 1

b) Utilisation de la transposition en avant pour assurer le respect de l'intention de l'usager 2, sur le site 1

Figure 1. *Respect de l'intention de l'usager*

La solution (Fig. 1.b) consiste donc à transformer l'opération à exécuter pour tenir compte des opérations concurrentes qui ont été sérialisées avant elle. Cette transformation est possible si l'on a défini au préalable une fonction, spécifique de la sémantique des opérations, qui donne pour chaque couple d'opérations concurrentes (op_1, op_2) une opération réalisant la même intention que op_2 mais à partir de l'état résultant de l'exécution de op_1. Cette fonction de transformation est appelée *transposition en avant* [Ellis et al, 1989].

La transposition en avant de l'opération op_2, pour tenir compte d'une opération concurrente op_1, nécessite que op_1 et op_2 soient générées à partir du même état de l'objet. Sans cela l'intention de l'usager peut être violée. Différentes solutions ont été proposées pour répondre à cette exigence. Dans [Suleiman et al, 1998] et [Sun et al, 1998] une nouvelle transformation, appelée *transposition en arrière*, est définie. Elle permet de changer l'ordre d'exécution de deux opérations consécutives tout en respectant leur intention.

Pour garantir la *convergence des copies*, la transposition en avant doit vérifier deux conditions. La première condition garantit que, partant d'un même état, l'exécution (sur un site) de op_1 suivie de la transposée de op_2 par rapport à op_1 produit le même état que l'exécution (sur l'autre site) de op_2 suivie de la transposée de op_1 par rapport à op_2. La seconde condition garantit que la transposition en avant d'une opération par rapport à une séquence d'opérations concurrentes ne dépend pas de l'ordre dans lequel les opérations de la séquence sont transposées. Il est toutefois possible de ne pas imposer cette condition et de se passer de la transposition en arrière, en mettant en œuvre un ordre global unique des opérations (par exemple au moyen d'un séquenceur).

D'une façon générale, le principe des algorithmes collaboratifs consiste à mémoriser pour chaque copie d'objet, l'histoire des opérations exécutées sur cette copie à partir d'un état initial pour produire l'état courant. Toute opération générée

localement est exécutée avant d'être ajoutée à l'histoire. Par contre, la réception d'une opération distante op nécessite une phase d'intégration qui consiste d'abord à transformer op par transposition pour déterminer l'opération op' réalisant la même intention que op, puis à exécuter op' sur l'état courant. Toutes ces méthodes se différencient par les traitements qu'elles font sur les histoires associées aux copies.

5.4. *Perspectives*

Synchroniseur générique

Dans la mesure où les techniques bâties autour des transformées opérationnelles permettent d'intégrer une opération (éventuellement retardée à la suite d'une déconnexion) dans l'histoire d'une copie et plus généralement de fusionner des histoires tout en respectant l'intention de l'usager et la causalité, elles sont particulièrement adaptées à la mobilité. En ce sens, nous pensons que le principe des transformées opérationnelles peut servir de modèle général pour la conception et la réalisation de Synchroniseurs génériques, adaptables à tous types d'objets et indépendants de leur structure, moyennant la fourniture des transposées appropriées.

Robustesse et passage à l'échelle des algorithmes multisynchrones

L'adaptation des algorithmes collaboratifs à la mobilité et au travail multi-synchrone (cf. 4.1) nécessite des recherches supplémentaires afin d'accepter dynamiquement un grand nombre d'utilisateurs et de rendre ces algorithmes tolérants aux défaillances, en particulier ceux qui utilisent un séquenceur (dont la perte peut être rédhibitoire) pour ordonner globalement les opérations.

Réconciliation et transformées opérationnelles

Les algorithmes collaboratifs synchrones sont par nature conçus pour tolérer une faible divergence des copies. Pour cette raison, des choix arbitraires peuvent être faits lors de l'écriture de certaines transposées en avant. C'est le cas, par exemple lorsque deux opérations concurrentes insèrent la même lettre, à la même position : on peut soit doubler la lettre, soit ne pas la répéter en considérant que cela correspond à la même intention. Si un utilisateur estime que son intention est violée par ce choix arbitraire, il peut réagir instantanément.

Au contraire, en autorisant les mises à jour lors des phases de déconnexion, les algorithmes multi-synchrones vont autoriser une plus grande divergence des copies. Dans ce contexte, pour faciliter la réconciliation ultérieure, voire mesurer la divergence [Molli et al, 2002], il est intéressant de signaler un conflit en cas de choix arbitraire dans le calcul d'une transposée. L'approfondissement des problèmes de réconciliation dans le cadre des transposées opérationnelles apparaît donc comme une piste de recherche majeure.

Compression et canonisation d'histoires

Une histoire, i.e. une séquence d'opérations associée à une copie, correspond en pratique à un journal (log). Dans le cadre de la mobilité, il est fréquent de transférer des histoires vers (ou à partir de) un site mobile qui se reconnecte. Pour cette raison, il est important de minimiser la taille d'une histoire de façon à réduire à la fois l'occupation mémoire (souvent critique pour un mobile) et la durée du transfert (faible bande passante). La transposition en arrière, en autorisant la permutation d'opérations consécutives, constitue une technique privilégiée pour compresser une histoire dans la mesure où elle permet de réduire le nombre d'opérations mises en jeu. Par exemple, deux opérations consécutives insérer(2,'abc') et supprimer(3) peuvent se résumer à insérer(2,'ac'). Des recherches portant sur la compression d'histoire, pour différents types d'objets, et la canonisation d'histoire (nombre minimal d'opérations disjointes) doivent être approfondies.

Formalisation de l'équivalence d'histoires respectant l'intention des usagers

Complétant ces études, des recherches plus théoriques portant sur la formalisation des histoires respectant la causalité et l'intention des usagers, et sur leur équivalence après transpositions, sont utiles. Le point de départ pourrait en être la théorie de la linéarisabilité [Herlihy et al, 1990].

6. Cohérence transactionnelle

6.1. *Motivation*

Comme toute application manipulant des données, une application mobile doit permettre l'accès concurrent et simultané à des données par des utilisateurs dispersés. Pour assurer la cohérence des données, les SGBDs classiques se fondent sur des mécanismes transactionnels. Il en est de même pour les systèmes de gestion de données permettant la mobilité, avec cependant des contraintes particulières quant à la réalisation de ces mécanismes transactionnels : certaines caractéristiques des applications mobiles rendent caduques les mécanismes classiques et doivent être prises en compte dans la conception de nouveau mécanismes.

6.2. *Position du problème*

Nous considérons des applications manipulant des données et intégrant des unités mobiles (UM) et au moins une unité fixe (UF) afin d'assurer la durabilité globale. Nous faisons également l'hypothèse que les accès aux données sont faits de manière transactionnelle. Les transactions sont initiées par des unités mobiles ou fixes, et sont potentiellement distribuées sur l'ensemble des unités. Les transactions peuvent être :

- exclusivement exécutées sur une ou plusieurs UM,

- demandées par une UM et exclusivement exécutées sur l'UF,
- exécutées partiellement sur une ou plusieurs UM et partiellement sur l'UF.

Si le contexte des applications mobiles est voisin de celui des BDs réparties, certaines caractéristiques essentielles lui sont propres : *les déplacements, les déconnexions et les durées des déconnexions*. Les déplacements peuvent impliquer des traitements particuliers du fait que le chemin d'accès aux UM change. Les *déconnexions* conduisent à reconsidérer certaines des approches existantes.

Pour de multiples raisons, et à tout moment, une UM peut se trouver déconnectée du réseau, de façon volontaire ou non. Cette déconnexion est totale dans le sens où toute communication avec les autres sites est impossible. *La durée de la déconnexion* n'est pas maîtrisée par l'application et dépend de contraintes externes (volonté de l'utilisateur, infrastructure). Cette durée n'étant pas bornée, le moment de reconnexion est donc imprévisible.

Ces caractéristiques spécifiques des environnements mobiles ont des conséquences importantes sur le support transactionnel. La déconnexion conduit à la création de copies des données sur l'UM afin de permettre l'exécution de transactions en mode déconnecté. La réintégration des copies dans la base de données initiale doit donc être prise en charge par le mécanisme de gestion des transactions. La durée non bornée de la déconnexion a des conséquences plus importantes encore, en particulier sur l'isolation. En effet, maintenir l'isolation, et donc conserver des données inaccessibles aux transactions concurrentes, peut devenir inacceptable au-delà d'un certain délai. D'un autre côté, la rupture de l'isolation conduisant à des modifications concurrentes et simultanées de données sur l'UM et sur l'UF rend difficile la réintégration des données après modifications locales. Cette réintégration est d'autant plus difficile lorsque la déconnexion est complète et permanente, empêchant ainsi l'échange de messages et/ou de données de synchronisation en avance.

6.3. *Solutions proposées*

De nombreux travaux cherchent à améliorer le support des transactions mobiles. [Serrano-Alvarado et al., 2001] propose une analyse de ces travaux, dont s'inspire ce paragraphe. Certains considèrent l'exécution des transactions dans l'ensemble alors que d'autres se focalisent sur des protocoles de validation, contrôle de concurrence (CC), ou autres points touchés par les caractéristiques des environnements mobiles. Comme mentionné précédemment, des protocoles pessimistes de CC tels que le verrouillage à deux phases ne sont pas appropriés en présence de déconnexions non bornées. Ainsi, [Momin et al., 2000] propose l'intégration d'approches optimistes et pessimistes en utilisant des temporisateurs et [Kim et al., 2001] propose deux algorithmes d'ordonnancement optimiste (0-Post et 0-Pre). Concernant la validation des transactions, certaines variantes du protocole de validation à deux phases ont été proposées. Elles cherchent à réduire le nombre de

messages et à tolérer la déconnexion de participants : [Bobineau et al., 2000] propose une validation unilatérale, [Kumar, 2000] et [Dunham et al., 1998] adoptent des temporisateurs.

Parmi la diversité de solutions considérant le support de transaction mobiles dans sa globalité, nous distinguons les propositions où les UM demandent des transactions qui s'exécutent sur le réseau fixe [Dunham et al. 1997, Yeo et al. 1994] de celles où les transactions s'exécutent complètement ou partiellement sur des UM [Pitoura et al. 1999, Gray et al. 96, Walborn et al. 1999, Walborn et al. 1995, Chrysanthis et al 1993, Madria et al. 2001]. Kangaroo [Dunham et al. 1997] et MDSTPM [Yeo et al. 1994] appartiennent au premier groupe et proposent un suivi de l'UM dans ses déplacements en offrant un support sur les stations de base les plus proches de l'UM. Les protocoles de CC et de validation utilisés sont classiques. Le second groupe de travaux traite les problèmes liés au stockage local (UM) et à la réintégration des données après manipulation locale. Clustering [Pitoura et al, 1999], Two-tier replication [Gray et al. 96], Semantics-based [Walborn et al, 1995], Promotion [Walborn et al, 1999] et Prewrite [Madria et al, 2001] permettent l'exécution des transactions localement avec une validation locale. La validation globale peut entraîner des mécanismes de réconciliation/synchronisation sauf quand les copies locales sont exclusives (e.g. Semantics-based, Prewrite). La sémantique des applications joue un rôle important lors de la réconciliation. Pour Clustering la divergence entre copies peut être bornée en limitant, par exemple, le nombre maximum de transactions exécutées ou le nombre de modifications faites sur une copie en mode déconnecté. Pour Two-tier replication, la validation globale consiste à ré-exécuter les transactions en prenant en compte certains critères d'acceptation (le résultat de la ré-exécution diffère de l'exécution locale). Pour Promotion, les données stockées localement sont encapsulées dans ce qu'ils appellent des compacts. Un compact contient, entre autres, des règles de cohérence, des méthodes spécifiques et des obligations. Si lors du traitement local les consignes définies dans le compact sont respectées alors la validation globale est garantie.

6.4. *Perspectives*

Les solutions proposées dans le cas de transactions s'exécutant sur une UM sont pour l'instant parcellaires : soit elles n'abordent que certains aspects, soit elles résultent de choix particuliers liés à des applications spécifiques. Il semble donc important d'aller vers des solutions plus générales intégrant les différentes dimensions du problème et capable de s'insérer dans différents champs d'application. Plus qu'un modèle de transactions spécifique résultant de choix particuliers, nous proposons de dégager un *canevas* permettant de situer différentes solutions et de mieux les analyser. Des éléments de définition d'un tel canevas sont proposés en 6.4.1 ; certains axes de recherche plus précis sont introduits en 6.4.2.

6.4.1 Un canevas pour les transactions s'exécutant en présence de déconnexions

Le *canevas* que nous proposons (voir Figure 2) se base sur des transactions souples de longue durée (TSL) s'exécutant sur des UM en mode connecté, déconnecté ou périodiquement connecté. Une transaction de ce type inclut une phase d'initialisation de l'environnement effectuée en mode connecté, suivie d'une phase d'exécution locale en mode éventuellement déconnecté dans laquelle les modifications sont réalisées au sein de transactions atomiques locales (TM_i). La TSL se termine par une phase de réintégration des données au moment de la validation, qui permet la synchronisation des modifications locales avec les éventuelles modifications concurrentes (TF_i). Lorsque c'est possible, la transaction peut utiliser des connexions temporaires au cours de son exécution pour transférer des informations de contrôle. Notons qu'une session de travail déconnectée peut comporter *plusieurs* TSL**.** Chaque phase d'une TSL peut avoir différentes réalisations ou faire l'objet de certains choix.

La **phase d'initialisation** de l'environnement comporte deux principales tâches : *création éventuelle de copies* sur l'UM et *établissement d'un contrat* entre l'UM et l'UF. Les copies créées peuvent être soit complètes, soit résumées (ou partielles) afin de correspondre à l'environnement de travail sur l'UM. Le contrat spécifie les contraintes d'exécution que l'UM doit respecter pour que l'UF lui garantisse des propriétés intéressantes lors de la phase de réintégration. Par exemple, le mode exclusif, où les données copiées demeurent verrouillées sur l'UF jusqu'à la réintégration, allège fortement celle-ci. Si le mode n'est pas exclusif, le contrat doit permettre (entre autres) de limiter la divergence entre les copies de l'UM et de l'UF. Le contrat peut aussi être évolutif : passé un certain seuil (ex. temporel), les contraintes d'exécution deviennent plus fortes sur l'UM (ex. restriction des opérations possibles sur les copies) et/ou les garanties offertes par l'UF diminuent, pour éviter de pénaliser les autres transactions, déléguant ainsi le maintien de la cohérence à la phase de réintégration.

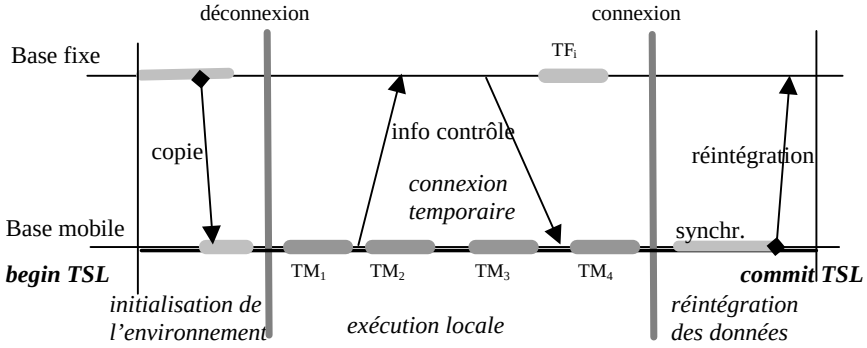


Figure 2. *Transactions Souples de Longue Durée (TSL)*

La **phase d'exécution** se fonde sur un moniteur transactionnel local garantissant certaines propriétés (e.g. ACID). Il assure également le suivi et l'évolution éventuelle (voire l'amendement) du contrat établi lors de la phase d'initialisation. Ce dernier point peut amener l'UM à effectuer des mesures de divergence sur l'UM et/ou l'UF et/ou à utiliser des re-connexions temporaires.

Les **phases de reconnexions** temporaires peuvent être opportunistes mais peuvent aussi être planifiées et donc forcées périodiquement par le contrat. L'information de contrôle échangée lors de ces reconnexions est diverse et dépend de la nature du contrat qu'elle aide à suivre. Elle peut aider, par exemple, au contrôle de concurrence (libérer ou acquérir des verrous), à l'amendement du contrat ou bien à limiter la divergence en intégrant à l'avance sur l'UM des mises à jour concurrentes déjà validées sur la base fixe pour faciliter la phase de réintégration.

La **phase de réintégration** a pour objet le retour cohérent des objets modifiés localement vers la base fixe. Elle suppose que l'UM est connectée de façon fiable au réseau fixe. Le déroulement de cette phase est très lié aux événements observés pendant les phases précédentes et se déroule en deux étapes. La première étape est consacrée à la détection de conflits induits par des actions concurrentes. La seconde étape est dédiée à la synchronisation proprement dite et peut offrir différents choix : abandon des modifications locales, écrasement des modifications sur la base fixe, fusion des copies, ré-exécution des transactions etc. Cette synchronisation utilise des algorithmes de mise en cohérence de copies (cf. section 5), mais doit également assurer la cohérence transactionnelle globale et donc tenir compte des dépendances et contraintes liant les objets modifiés.

6.4.2 Axes de recherche

Le contrat passé entre l'UM et l'UF est la notion centrale du canevas présenté ci-dessus. Le premier axe de recherche est d'approfondir cette notion, plus particulièrement en développant les points suivants.

- *Définition du contrat* : il s'agit de préciser les éléments constitutifs du contrat (type d'isolation, seuils de divergence, seuils temporels, planification des reconnexions temporaires,…) ainsi que le modèle pour les assembler et permettre aux utilisateurs de concevoir finement la stratégie d'exécution de leur transaction mobile.
- *Mise en œuvre du contrat* : il s'agit de définir quels composants (données et programmes) doivent être embarqués sur l'UM afin de suivre le contrat, en tenant compte des limitations en capacité de l'UM.
- *Achèvement du contrat* : le respect du contrat par l'UM doit entraîner une réintégration « optimisée », en choisissant l'algorithme de synchronisation existant le mieux adapté au contrat. De plus, la plupart de ces algorithmes travaillant « objet par objet », il est nécessaire d'inclure un mécanisme permettant de vérifier la cohérence globale de la TSL après réintégration.

- *Comportement de l'UF* : l'UF étant partie prenante du contrat, il convient d'étudier le comportement de l'UF lors des différentes phases d'une transaction TSL afin d'offrir les garanties spécifiées par le contrat.

Un autre axe de recherche connexe concerne la présence de contraintes d'intégrité (CI). En effet, la vérification des CI peut impliquer des données non directement touchées par la transaction mobile et qui ne peuvent pas être embarquées facilement sur l'UM (contraintes de place et de disponibilité). Le report de la vérification à la phase de synchronisation, peut entraîner l'invalidation tardive de beaucoup de travail fait sur l'UM. Il serait donc intéressant de prendre en compte les CI lors des trois phases du canevas, afin d'établir la meilleure stratégie pour leur vérification.

7. Gestion de données embarquées

7.1. *Motivations*

Les calculateurs ultra-légers revêtent aujourd'hui de multiples formes (PDA, téléphones cellulaires, cartes à puce, décodeurs intelligents, puces pour la domotique, l'automobile ou l'avionique) pour satisfaire des besoins applicatifs de plus en plus variés. L'évolution des architectures de ces calculateurs et de leurs usages font s'accroître le besoin d'embarquer des données et des composant base de données.

Par exemple, les réseaux de capteurs utilisés pour collecter des informations environnementales (météorologie [ALERT], pollution, trafic routier [SIRIUS]) sont en train d'évoluer vers de véritables bases de données distribuées où chaque capteur devient un nœud *actif* du système; i.e., un micro-serveur de données interrogeable à distance. Les capacités locales de traitement permettent d'agréger, de trier ou filtrer les données et d'économiser ainsi du trafic réseau, particulièrement coûteux dans une infrastructure sans-fil.

La gestion de données personnelles sur des puces est une autre motivation des traitements embarqués. La carte à puce est par exemple utilisée dans plusieurs applications manipulant des informations personnelles et confidentielles (e.g., santé, assurance, téléphonie, etc.). Dans ces contextes, les traitements peuvent être relativement complexes (e.g., sélections, jointures, agrégation) et doivent être confinés dans la puce pour ne révéler aucune donnée confidentielle.

Enfin, les ordinateurs de poche comme les ordinateurs de bord équipant certains véhicules permettent de charger des données pouvant être interrogées en mode déconnecté (e.g., fragment d'une BD d'entreprise, information touristique, etc.). Dans ce contexte, les traitements peuvent être aussi complexes que dans des bases de données classiques.

Ainsi, l'économie des coûts de communication, la confidentialité des données et le support de traitements déconnectés sont trois motivations importantes conduisant à exécuter des traitements embarqués sur des calculateurs légers.

7.2. *Problématique*

La prise en compte des architectures des calculateurs légers, souvent très spécifiques, nécessite de revoir en profondeur les techniques et algorithmes utilisés pour la manipulation des données embarquées. Les principaux paramètres de ces architectures influant sur la gestion des données embarquées sont :

- La taille et la technologie des composants de mémoire stable (e.g., EEPROM, Flash, RAM alimentée, etc.) destinés a accueillir les données embarquées. Le type de mémoire définit le temps d'accès, la granularité de ces accès ainsi que le nombre maximum d'accès en écriture [Dipert, 1997];
- La taille de la mémoire de travail (RAM) utilisée comme mémoire tampon lors de la manipulation des données;
- La vitesse du processeur et sa complexité (i.e., possède-t-il des caches instructions et des caches de données ?);
- La connexion au réseau et le type de réseau considéré (e.g., réseau ad-hoc sans fil, connexion point à point au réseau filaire, etc.);
- L'autonomie électrique du calculateur souvent peu abondante (e.g., capteur), voire inexistante hors connexion (e.g. carte à puce).

En fonction de ces paramètres, nous déclinons ci-dessous les principaux problèmes induits sur la gestion des données embarquées. Nous ne considérons pas les problèmes de synchronisation, de cohérence transactionnelle ni de confidentialité des données embarquées, qui sont traités respectivement dans les sections 5, 6 et 8.

- *Stockage des données :* La mémoire stable destinée aux données embarquées étant souvent de faible capacité, il est nécessaire de proposer des modèles de stockage les plus compacts possibles. D'autre part, les caractéristiques spécifiques des mémoires stables utilisées peuvent nécessiter la conception de modèles de stockage ad-hoc. Par exemple, l'utilisation de mémoire Flash nécessitera des mises à jour par « bancs » afin d'éviter des performances d'écriture désastreuses. Remarquons que la définition d'un modèle de stockage adapté dépend aussi de la fréquence respective des consultations et des mises à jour.
- *Interrogation locale:* Les données embarquées peuvent être interrogées localement, c'est à dire en ayant recours uniquement aux ressources du calculateur léger. Les algorithmes d'évaluation de requêtes doivent alors être revus pour pouvoir fonctionner avec une quantité (très) réduite de mémoire de travail. La définition de structures d'indexation peut éventuellement contribuer à cet objectif. Cependant, ces structures d'indexation doivent être être compactes et adaptées au type de mémoire stable du calculateur. Enfin, des contraintes de consommation électrique peuvent venir encore complexifier le problème.
- *Interrogation distante de plusieurs calculateurs :* L'interrogation de plusieurs calculateurs pose des problèmes nouveaux en termes de langage d'interrogation et de stratégies d'exécution distribuée entre les calculateurs légers et le poste

(éventuellement) fixe d'où a été émise la requête. Au niveau du langage, il faut pouvoir spécifier quels sont les calculateurs et les données concernées par la requête notamment dans le cas de capteurs acquérant des données en continu. Au niveau de l'exécution, il faut prendre notamment en compte les contraintes concernant le réseau afin de répartir l'exécution entre les différents calculateurs.

7.3. *Solutions proposées*

Dans cette section, nous distinguons les solutions *généralistes*, i.e., fonctionnant sur un ensemble de calculateurs mobiles, des solutions ayant pour cible un type particulier de calculateur et prenant donc en compte ses contraintes spécifiques.

- *Solutions généralistes :* Les éditeurs de SGBD ont adressé le problème de la gestion de données embarquées en proposant des versions «légères» de leur SGBD, comme SQL Anywhere Studio [Giguère, 2001], Oracle 8i light [Oracle, 2002], SQLServer pour Windows CE [Seshadri et al, 2000] ou DB2 Everyplace [Karlsson et al, 2001]. Ces produits ont été conçus principalement pour des ordinateurs portables et des assistants personnels. Ils ont une petite empreinte mémoire obtenue en simplifiant et en modularisant le code du SGBD. Par exemple, SQLServer Studio permet de conserver uniquement le code correspondant aux fonctionnalités utilisées dans l'application embarquée. Les éditeurs de SGBD se sont surtout focalisés sur l'aspect déploiement et synchronisation des données, privilégiant une portabilité (multi-plateforme) importante à des techniques spécifiques à tel ou tel calculateur. Ainsi, à notre connaissance, peu de travaux ont été entrepris pour adapter les techniques de stockage, d'indexation ou d'interrogation aux spécificités des calculateurs.
- *Solutions sur carte à puce :* La carte à puce est indubitablement le plus portable, le plus sûr et le moins cher des calculateurs. Ces caractéristiques ont motivé des travaux depuis quelques années sur ce support. Le premier standard de bases de données pour cartes à puce, appelé SCQL [ISO, 1999], a été publié en 1999, suite à des travaux initiés au laboratoire LIFL. Ce standard est limité à des opérations simples comme la sélection mono-table, en raison des contraintes très fortes sur la capacité mémoire des cartes à puce de la fin des années 90. Le SGBD SQLJava Machine de ISOL [Carrasco, 1999], proche du standard SCQL, constitue la première tentative commerciale de SGBD sur carte à puce. Un moteur de SGBD embarqué, appelé PicoDBMS, supportant des requêtes Select-Project-Join avec agrégation en complément des mécanismes traditionnels de vues et de droits a été plus récemment proposé [Pucheral et al, 2000]. PicoDBMS est basé sur un modèle de stockage compact et efficace, tirant avantage de l'accès direct en EEPROM par l'utilisation extensive de pointeurs. Le point central de cette architecture est un évaluateur de requêtes capable de traiter tout type de requête SQL sans consommation de RAM, quel que soit le volume de données interrogé [Anciaux et al, 2001]. La mise en œuvre de protocoles transactionnels dédiés aux cartes à puce a également fait l'objet d'études [Lecomte et al, 1997].

- *Solutions incluant des capteurs :* Les membres du projet COUGAR de l'Université de Cornell [Bonnet et al. 2000, Bonnet et al. 2001] ont défini le concept de «Device Database Systems», une base de données distribuée incluant un ensemble de capteurs participant à l'exécution de requêtes ad-hoc. Cette approche est opposée à l'approche classique où l'ensemble des capteurs émettent les données vers un SGBD central. La motivation est double : (i) les capteurs peuvent acquérir l'information utile (e.g., la mesure que doit faire le capteur peut être indiquée dans la requête); et (ii) les capteurs ne transmettent que l'information utile en filtrant ou agrégeant les données afin d'économiser la bande passante réseau. Cette approche a été étendue par l'Université de Berkeley à la gestion de requêtes agrégatives sur un ensemble de capteurs [Madden et al, 2002 (a)] et de requêtes sur des flux de données émises par les capteurs [Madden et al, 2002 (b)]. Des extensions sont apportées au langage SQL pour prendre en compte la localisation des capteurs, la fréquence d'échantillonnage ainsi que la durée de la requête.

7.4. *Perspectives*

Actuellement, les éditeurs de SGBD proposent des solutions généralistes et pragmatiques mais souvent sous-optimales et inadaptées à des calculateurs très fortement contraints. Les études académiques se focalisent elles sur des solutions ad-hoc prenant en compte les contraintes spécifiques de certaines architectures (e.g., cartes à puce, capteurs). L'étude systématique de modèles d'évaluation et d'optimisation de requêtes pour des architectures fortement contraintes constitue un challenge de premier ordre pour les années à venir. L'objectif visé est d'intégrer dans un même modèle ces différentes contraintes afin d'en déduire des règles de conception pour des moteurs bases de données dédiés ainsi que des recommandations pour les concepteurs de ces plate-formes matérielles. Des travaux de co-conception, mentionnés comme une des priorités de l'appel d'offre RNTL'03, pourraient ainsi permettre de spécifier le couple logiciel/matériel le mieux à même de réaliser un certain traitement en fonction d'un ensemble de contraintes (taille, coût, temps de réponse, etc.). Dans cette optique, il est essentiel de comprendre l'influence de chaque contrainte sur l'évaluation de requêtes mais aussi l'interaction entre les différentes contraintes. La définition de modèles de coût et de bancs d'essais adaptés est indispensable à de telles études.

Enfin, remarquons que les travaux existants considèrent tous le modèle relationnel pour les données embarquées. L'émergence de XML comme standard d'échange de données entre différents systèmes doit être pris en compte dans le stockage et l'interrogation de données embarquées.

8. Confidentialité des données

8.1. *Motivations*

La préservation de la confidentialité est un enjeu majeur dans la majorité des applications, qu'il s'agisse de commerce électronique, de gestion de données personnelles, de systèmes d'information en ligne ou de préservation de secrets industriels, scientifiques ou militaires. Cette problématique se pose avec une acuité particulière dans le contexte de la mobilité. En effet, de plus en plus de données personnelles (dossiers portables) sont stockées sur des calculateurs mobiles (PDA, téléphones cellulaires, cartes à puce …) et la confidentialité de ces données peut être menacée si le mobile est perdu ou volé. Par ailleurs, un utilisateur mobile peut avoir recours à un *Database Service Provider* (*DSP*) qui lui assurera un accès permanent et fiable à ses données via l'Internet, sans limite de volume. Le fait que les données soient dans ce cas hébergées par un tiers dans lequel il est légitime de ne pas fonder une confiance totale introduit une nouvelle menace envers la confidentialité. Le problème se complique encore lorsque les données ont vocation à être partagées par plusieurs utilisateurs. Le partage peut se produire pour des dossiers portables (ex : dossier médical sur carte à puce accédé par le patient, les médecins, les pharmaciens, l'organisme de sécurité sociale) comme pour des données stockées chez un DSP (ex : documents, plans, contrats partagés par une équipe de collaborateurs mobiles). Pour contrôler le partage, chaque utilisateur doit être identifié et des droits précis doivent lui être associés. L'identification et l'affectation de droits à des utilisateurs devient un problème en soi dans certaines situations liées à la mobilité : réunions de travail entre utilisateurs formant un groupe dynamique (des membres joignent ou quittent le groupe dynamiquement), diffusion sélective d'informations à destination d'une population d'utilisateurs mobiles pouvant avoir des droits différents.

8.2. *Problématique*

La Figure 3 symbolise les différents modes d'accès aux données à partir de mobiles selon que les données sont partagées ou non et selon qu'elles sont stockées sur un mobile ou sur un serveur.

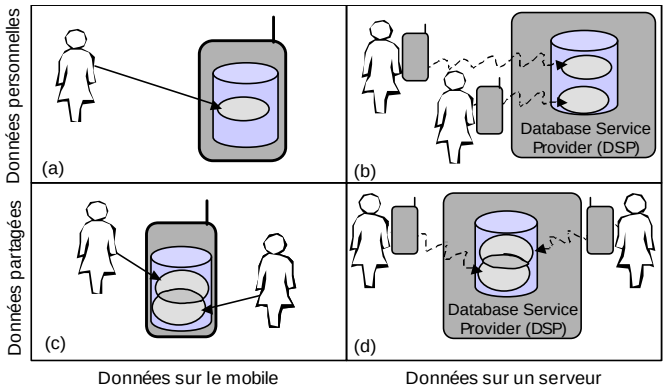


Figure 3. *Scénarios d'accès aux données*

Dans chacun de ces scénarios, différentes menaces pèsent sur la confidentialité des données :

- *Interception des communications :* un intrus capte les communications entre un client et un serveur (scénarios b et d) et accède illégalement aux données confidentielles contenues dans les messages. Cette menace est d'autant plus réelle dans un environnement mobile qu'il est très facile d'intercepter des communications hertziennes.
- *Usurpation d'identité :* un intrus s'identifie à la place d'un utilisateur répertorié par le système et accède ainsi à ses données (tous les scénarios).
- *Outrepassement de droits :* un utilisateur régulièrement identifié par le système tente d'accéder à des informations pour lesquelles il n'a pas les autorisations nécessaires (scénarios b, c et d). Cette attaque peut être menée par un utilisateur qui déduit une information non autorisée par une combinaison astucieuse de requêtes autorisées. Cette attaque peut par ailleurs être très facilement menée par un administrateur de bases de données (où par toute personne usurpant son identité) qui possède par définition tous les droits.
- *Analyse de l'empreinte disque de la base de données :* un intrus subtilise le mobile d'un utilisateur (scénarios a et c) ou bien s'infiltre sur le serveur d'un DSP (scénarios b et d) et en extrait l'empreinte disque de la base afin de récupérer des informations confidentielles en court-circuitant le gestionnaire de droits du SGBD gérant la base.
- *Détournement d'informations détenues légalement :* tout DSP détient légalement des informations sur ses clients (identité, adresse, informations bancaires, statistiques sur l'utilisation du service offert) et est tenu de respecter une charte de confidentialité relative à la détention et à l'utilisation de ces informations. Cette charte ne représente qu'un engagement moral difficilement contrôlable.

8.3. *Solutions proposées*

Cette section présente les solutions apportées dans l'état de l'art à chacune des menaces identifiées précédemment.

- *Interception des communications :* le moyen le plus courant pour sécuriser une communication est de faire appel à des techniques de chiffrement, comme c'est le cas par exemple des protocoles SSL (Secure Socket Layer) ou TLS (Transport Layer Security). Le chiffrement du message garantit la confidentialité du contenu échangé et peut être complémenté par des techniques de hachage assurant l'intégrité du message et de signature assurant la non-répudiation des messages [Schneier, 1996].
- *Usurpation d'identité :* tout utilisateur d'un SGBD doit être identifié (il doit dire qui il est) et authentifié (il doit le prouver). L'authentification peut se faire par le biais d'un mot de passe, éventuellement relayé par un serveur d'authentification (e.g., Kerberos) qui évite le cheminement en clair du mot de passe sur le réseau.

L'authentification peut être renforcée par des techniques matérielles (cartes à puce, token card …) ou biométriques [Oracle, 1999].

- *Outrepassement de droits :* la gestion des droits d'accès dans les SGBD peut suivre différents modèles [Baraani et al, 1996]. Le modèle de droits défini dans SQL2 est de type DAC (Discretionary Access Control), c'est à dire que des utilisateurs se voient affecter le droit d'effectuer certaines actions (lectures, modifications) sur certains objets (tables, vues, procédures) et peuvent transmettre certains de leurs droits à d'autres utilisateurs. Le modèle RBAC (Role-Based Access Control), supporté par SQL3, affecte des droits à des rôles et un utilisateur peut être autorisé à jouer des rôles différents au cours de sessions différentes. Enfin, le modèle MAC (Mandatory Access Control), supporté notamment par Oracle8i (Oracle Label Security), fixe des niveaux hiérarchiques de sécurité aux données (public, confidentiel, secret …) et des niveaux d'habilitation aux utilisateurs. Cette palette de modèles couvre bien les besoins de la majorité des applications et sont difficiles à contourner. Par contre, il reste difficile de se prémunir contre un utilisateur malicieux recoupant les résultats de requêtes autorisées pour déduire une information qui lui est cachée. Seule une politique d'audit des requêtes émises par chaque utilisateur permet de détecter ce type d'attaques. Par ailleurs, le modèle de droits le plus restrictif reste inopérant contre l'administrateur de la base de données (DBA) dont un des privilèges est justement de gérer les droits. Les rares solutions apportées à ce problème reposent sur le chiffrement des données des utilisateurs. Ce point est détaillé dans la section suivante.
- *Analyse de l'empreinte disque de la base de données :* Ce type d'attaques peut être évité si la base de données est hébergée sur un support sécurisé. C'est le cas par exemple des dossiers portables embarqués sur des cartes à puce [Pucheral et al, 2000]. Dans le cas contraire, diverses techniques de chiffrement des données permettent de se prémunir contre des attaques dirigées par un intrus sur l'empreinte disque de la base de données ou des attaques dirigées par un DBA. Oracle propose un package PL/SQL permettant d'effectuer du chiffrement/déchiffrement à clé secrète [Oracle, 1999]. La gestion des clés de chiffrement est laissée à la charge de l'application utilisant ce package. Cette solution est efficace contre un intrus mais pas contre un DBA qui a le pouvoir de modifier le package de chiffrement et d'intercepter les clés ou les données en clair lors du déchiffrement (les données étant déchiffrées sur le serveur lors de l'évaluation des requêtes). Une première solution a ce problème consiste à modifier le noyau du SGBD pour interdire certaines actions du DBA [He et al, 2001]. Une seconde solution est de distinguer les fonctions d'administration des données des fonctions d'administration de la sécurité et d'isoler ces dernières dans un serveur séparé contrôlant les droits d'accès et les clés de chiffrement [Mattsson, 2000]. Ces différentes solutions souffrent du fait que les données sont toujours déchiffrées sur le serveur et permettent diverses attaques du DBA. L'alternative est de réaliser le déchiffrement des données sur le client. Ceci implique de décomposer toute requête en une sous-partie exécutable sur le

serveur sur les données chiffrées et une seconde partie exécutable uniquement sur le client après déchiffrement (prédicats d'inégalité, agrégation, tri …) [Hacigumus et al, 2002]. Le problème se complique dès que les données sont partagées par plusieurs clients ayant des droits d'accès différents [Bouganim et al, 2002].

- *Détournement d'informations détenues légalement :* de nombreux cas de violation de chartes de confidentialité, pourtant publiées sur les sites Web des prestataires de services Internet, ont été relevés [Olsen, 1999]. Le concept de *SGBD Hippocratique*, à savoir de SGBD donnant l'assurance du respect d'un serment de confidentialité, a été introduit dans [Agrawal et al, 2002]. Un tel SGBD se doit de respecter un ensemble de principes fondateurs parmi lesquels : préciser l'objectif d'utilisation de chaque donnée collectée sur un utilisateur et recueillir l'assentiment de l'utilisateur sur cet objectif, ne stocker que l'information strictement nécessaire à l'atteinte de cet objectif et uniquement pendant le laps de temps strictement nécessaire, ne pas divulguer cette information à des tiers sans autorisation préalable de l'utilisateur, donner à l'utilisateur la possibilité de consulter les informations qui le concernent et enfin offrir des outils permettant à un tiers de contrôler que ces principes sont bien respectés. Le standard P3P promu par le W3C [Marchiori, 2002] est également un moyen donné aux utilisateurs pour mieux contrôler les informations que les sites Web accumulent sur eux.

8.4. *Perspectives*

Un travail considérable d'investigation reste à mener autour des techniques de chiffrement de bases de données et des techniques d'exploitation (interrogation, mises à jour) de telles bases de données. Compte tenu du volume de données chiffrées et de leur persistance, il est impératif pour les cryptographes de définir des algorithmes de chiffrement résistants aux attaques statistiques. Par ailleurs, des techniques de chiffrement autorisant l'application d'un ensemble d'opérations arithmétiques sur les données chiffrées [Domingo-Ferrer, 1997] permettraient d'élargir la portée des requêtes traitables par un serveur de données chiffrées et réduiraient d'autant le coût des méthodes assurant le déchiffrement sur le(s) client(s). Le partage des données dans un tel contexte reste un problème majeur. Ce partage n'est possible que si le contrôle des droits d'accès est assuré sur le client par un équipement sécurisé [Bouganim et al, 2002], par exemple une carte à puce. De nouvelles formes de processeurs sécurisés [Smith et al. 1999, iButton 2002] devraient permettre de s'affranchir des contraintes inhérentes aux cartes à puce et pourraient ouvrir la voie vers de nouvelles architectures pour accéder et partager des données chiffrées.

Si les modèles de droits d'accès, tels que DAC, RBAC et MAC sont aujourd'hui bien établis, il faut être capable de les instancier dans le contexte particulier de la mobilité. Par exemple, comment définir un modèle d'échange de données sécurisé entre utilisateurs formant un groupe dynamique dont les membres peuvent joindre

ou quitter le groupe à tout moment ? Comment définir un modèle de diffusion sélective d'informations à destination d'une population d'utilisateurs mobiles ayant des droits différents sur les données diffusées ? Le problème se complique lorsque l'entité gérant le stockage des données, leur partage et leur diffusion n'est pas un tiers de confiance (par exemple, un Database Service Provider). Des éléments de réponse sont apportés par [Devanbu et al, 2000] mais concernent plus la vérification de la complétude et de l'authenticité des données émises par un DSP que la préservation de la confidentialité de ces données.

Enfin, si les préceptes d'un SGBD hippocratique ont été énoncés, il reste à mettre en œuvre de tels systèmes. Ceci passe notamment par la définition de nouveaux noyaux de SGBD capables d'intégrer les règles de confidentialité établies avec chaque utilisateur au sein même de l'évaluation des requêtes et offrant des outils d'audits non falsifiables. Un problème connexe est la mise en œuvre de techniques d'interrogation de données ne permettant pas au serveur d'identifier le sous-ensemble d'informations accédé par un utilisateur [Cachin et al, 1999].

9. Adaptabilité des applications et des ressources

9.1. *Motivations*

Nous considérons des applications « classiques » d'accès aux données, dans lesquelles des utilisateurs accèdent en lecture et/ou en écriture, via un réseau de communication, à des bases de données stockées sur un ou plusieurs serveurs distants. Avec le nomadisme croissant des utilisateurs et le développement de terminaux portables dotés de connexion réseau sans fil (GSM et/ou WiFi), de nouveaux besoins apparaissent. L'utilisateur doit pouvoir, quelle que soit sa localisation géographique et le type de terminal dont il dispose, accéder à ses applications et obtenir un service aussi proche que possible de celui fourni dans un contexte statique, où le terminal de l'utilisateur est fixe, dispose de ressources importantes (puissance du processeur, capacités d'affichage et de stockage) et est relié aux serveurs par un réseau fiable et performant. Par exemple, un ingénieur commercial peut souhaiter mettre à jour, chez un client ou dans le train au retour de sa visite, la base des stocks de l'entreprise en fonction de ses ventes de la journée. Il s'agit donc d'**adapter** des applications classiques, et/ou la gestion des ressources système et réseau qui supportent ces applications, au contexte nouveau de la mobilité.

9.2. *Position du problème*

L'environnement d'une session d'application d'accès aux données peut être défini comme l'ensemble des conditions matérielles et logicielles dans lesquelles se déroule la session applicative. Dans un contexte de mobilité (au double sens du nomadisme des utilisateurs et de la mobilité des terminaux), cet environnement est caractérisé par une grande variabilité :

- les utilisateurs peuvent accéder à leurs applications depuis un lieu fixe quelconque, via un terminal fixe de nature quelconque (« macro-mobilité ») ; ceci peut impliquer l'accès à des données dépendantes de la localisation (services de proximité par exemple), la prise en compte de profils utilisateur différents (par exemple, au bureau et au domicile), la découverte dynamique des services accessibles, et la mise en œuvre de niveaux de sécurité différents ;
- les terminaux portables ont, par rapport aux terminaux fixes de même génération, des ressources limitées (taille de l'écran, capacité de traitement et de stockage, énergie) impliquant l'adaptation de la présentation des données aux caractéristiques de l'interface utilisateur, ainsi qu'une éventuelle redistribution des fonctionnalités de l'application entre la partie qui s'exécute sur le terminal et celle qui s'exécute sur des machines fixes ;
- les liaisons sans fil sont caractérisées par un coût élevé (sinon financier, du moins en énergie), une bande passante réduite et variable dans le temps, et sont sujettes à des déconnexions impromptues (imperfections du réseau) ;
- la variation dynamique de l'environnement est encore plus accentuée lorsque l'utilisateur muni d'un terminal portable se déplace en cours d'exécution de l'application (« micro-mobilité ») : passage d'un réseau physique à un autre de caractéristiques différentes (UMTS), connexions temporairement impossibles (par exemple, passage dans une zone d'ombre, ou interdiction d'usage en avion).

Il s'agit donc d'adapter les applications et/ou la gestion des ressources utilisées par les applications aux variations dynamiques de l'environnement inhérentes à la mobilité. D'une part, la relative pauvreté des ressources des terminaux portables plaide pour s'appuyer autant que possible sur les serveurs fixes ; d'autre part, la non-fiabilité et les mauvaises performances des réseaux sans fil, ainsi que le coût des communications, poussent à favoriser le fonctionnement autonome [Satyanarayanan, 1996].

9.3. *Solutions proposées*

Une discussion complète des nouveaux paradigmes pour les interactions client-serveur en environnement mobile, dont l'adaptation fait partie, peut être trouvée dans [Jing et al., 1999]. Des solutions ont été proposées dans la littérature pour répondre partiellement aux nécessités d'adaptation dynamique décrites ci-dessus ; les travaux les plus significatifs sont les suivants.

Adaptation à la variabilité de la bande passante des réseaux sans fil

Le système Odyssey introduit la notion de « fidélité » des données. Celles-ci sont disponibles sur les serveurs en plusieurs degrés de précision ; au moyen d'une API particulière, les applications peuvent commuter dynamiquement d'un niveau de fidélité à un autre en fonction de la bande passante réseau disponible [Noble et al., 1997] ou de l'énergie restante [Flinn et al., 1999]. Les services considérés sont

limités à la consultation de données multimédias (images, documents vidéo ou cartographiques). L'adaptation ne concerne que le côté client.

Adaptation des applications au mode déconnecté

La boîte à outils Rover [Joseph et al., 1997] supporte les déconnexions réseau par deux mécanismes complémentaires : la structuration des applications en objets « relogeables » et un mécanisme de RPC asynchrone. En mode déconnecté, le client accède (en lecture ou en écriture) à une copie locale des objets serveur ; les modifications conflictuelles effectuées sur la copie primaire et sur la copie locale sont détectées pendant les phases de connexion, la résolution des conflits étant laissée à l'application. L'adaptation concerne les côtés client et serveur. Les applications doivent être programmées selon un modèle propriétaire d'objets relogeables, ce qui limite l'adaptation d'applications existantes. Le système DOM [Conan et al., 2002] reprend les concepts de Rover pour des applications distribuées à objets conformes aux spécifications CORBA.

Adaptation des couches réseau à la micro-mobilité

Les protocoles de l'Internet ont été conçus pour les environnements filaires et s'adaptent mal aux environnements sans fil, d'une part à cause de la qualité médiocre de l'environnement de propagation et d'autre part, à cause de la mobilité de l'utilisateur. Deux catégories de mobilité se distinguent [Perkins, 1996].

La macro-mobilité représente la possibilité donnée à un utilisateur d'accéder à ses services indépendamment de sa localisation Certains travaux comme *Mobile-IP* répondent à ce besoin selon le principe suivant. Tout nœud mobile possède une adresse dans son réseau d'origine et fait une demande d'adresse temporaire dans le réseau visité. Le routage des messages du réseau d'origine vers le réseau visité se fait via une table de correspondance associant les deux types d'adresse. Cette solution permet à l'utilisateur de se connecter au réseau IP sans intervention manuelle. Par contre elle procure un niveau de sécurité faible, car un terminal qui utilise une adresse IP d'un autre terminal peut intercepter toutes les données destinées à ce dernier.

La micro-mobilité permet à un utilisateur de rester connecté au réseau à tout moment. Ce principe nécessite deux mécanismes. Le premier est la localisation de l'utilisateur, qui à partir de la base de données du réseau et de la signalisation échangée entre le terminal et le réseau, permet de déterminer à chaque instant la cellule où le mobile est connecté. Le second est la conservation de la communication lorsque le mobile passe d'une cellule à une autre.

Avec ces deux types de mobilité l'adaptation des couches basses aux couches hautes est un des problèmes les plus importants pour fournir un niveau de qualité de service aux applications. Ces critères de qualité de service sont :

- la bande passante qui peut varier de quelques Kbit/s à quelques Mbit/s,

- la qualité du lien radio qui peut être mesuré par le rapport signal sur bruit,
- le délai de transit de bout en bout pour des applications qui ont des contraintes temporelles comme le multimédia ou la voix,
- la probabilité de perte pour des applications de données.

L'adaptation des couches basses du réseau au besoin de qualité de service dépend du type de mobilité. Dans les systèmes micro-mobiles, des politiques d'allocation de ressources basées sur des techniques à étalement de spectre permettent d'avoir une meilleure utilisation de la bande passante. Dans les systèmes macro-mobiles, des solutions basées sur les seuils sont généralement proposées (i.e., au delà d'un certain seuil, seuls les clients privilégiés sont acceptés). Plusieurs types de flux peuvent être distingués [legrand et al, 2000] : flux prioritaire destiné aux applications nécessitant un certain niveau de qualité de service, flux non prioritaire pour des applications de type best effort. Assurer la continuité de service lors du passage d'une cellule à une autre nécessite des mécanismes de réservation de ressources dans les cellules voisines de celle où se trouve un client privilégié.

9.4. *Perspectives*

L'approche suivie dans Odyssey nous semble la plus prometteuse en matière d'adaptation dynamique des applications et de la gestion des ressources aux fluctuations de l'environnement dans un contexte de mobilité. Elle repose sur une collaboration entre les applications et le système (au sens large) sous-jacent. Les applications sont les mieux placées pour déterminer comment s'adapter aux variations de l'état de l'environnement (*politiques d'adaptation*) ; le système est le mieux placé pour observer et notifier ces variations (*mécanismes d'adaptation*). Par ailleurs, la notion de « fidélité », que l'on peut définir de manière générale comme le degré de similarité entre les données vues par le client mobile et les données réelles, est utilisée par les applications pour ajuster leur mode de fonctionnement aux possibilités actuelles de l'infrastructure sous-jacente. Cette notion, pivot des stratégies d'adaptation, peut être exploitée de manière beaucoup plus riche qu'elle ne l'est dans Odyssey, où elle n'est utilisée que pour représenter la qualité de représentation de documents multimédias ou cartographiques. Nous nous proposons d'aborder le problème de l'adaptation dynamique sous deux aspects.

Représentation et expression de la résolution des données

Dans un contexte d'accès mobile à des bases de données, la résolution (qualité) des données vues par le client peut être représentée selon différentes dimensions, notamment : précision du contenu (fidélité de représentation au sens d'Odyssey, mais aussi degré de fragmentation – horizontale ou verticale – des relations d'une base de données) ; cohérence temporelle (degré de « fraîcheur » des données vues par rapport aux données réelles) ; cohérence sémantique (degré de respect des contraintes d'intégrité) ; niveau de sécurité. Ces dimensions doivent pouvoir être

combinées, pour fournir autant d'axes sur lesquels peut porter la stratégie d'adaptation propre à une application ou un utilisateur.

La première difficulté est de déterminer comment décrire un niveau de résolution multi-dimensionnelle, la valeur sur chaque dimension n'étant pas toujours immédiatement quantifiable. La seconde est d'exprimer, dans l'espace des résolutions, celles qui sont pertinentes : au niveau du schéma de la base, seuls certains niveaux de résolution sont possibles ; pour une application utilisant ces données, seul un sous-ensemble est valide ; enfin, pour une exécution particulière d'une application, seul un sous-ensemble des résolutions valides peut être pertinent.

La résolution peut être aussi utilisée pour faire du « rechargement de données » (« *data recharging* »). Par analogie au rechargement des batteries, cela consiste à rafraîchir les données présentes sur le client en fonction de la durée de la connexion (plus on reste connecté longtemps, meilleur sera le rafraîchissement des données). Pour un temps de chargement donné, il faut donc déterminer quelle est la charge optimale. Ceci est bien évidemment dépendant de l'importance des données, celle-ci pouvant être estimée via un profil utilisateur donnant des informations sur la pertinence des données. Des langages d'expression de profils ont déjà été proposés [Cherniak et al, 2001] mais manquent de généralité et ne prennent pas en compte la résolution des données.

Expression des demandes applicatives et architecture d'adaptation dynamique

La séparation des politiques et des mécanismes peut être réalisée au moyen d'une couche d'adaptation, entité logicielle s'exécutant sur chaque terminal mobile interposée entre le code applicatif et le système sous-jacent. Dans le sens descendant, cette couche d'adaptation reçoit les demandes applicatives en terme de qualités de service, les traduit en requêtes sur les ressources concernées (réseau, base de données, …) et fournit à l'application le « meilleur » mode de fonctionnement en fonction de l'état courant des ressources. Dans le sens ascendant, la couche d'adaptation (qui supervise en permanence l'état des ressources) notifie de façon asynchrone à l'application les changements d'état « significatifs » nécessitant le passage à un nouveau mode de fonctionnement – ou bien effectue elle-même, suivant un schéma pré-défini par l'application, ce passage. C'est la démarche suivie par Odyssey, mais dans le cas simple où le seul degré de liberté concerne le niveau de fidélité d'un document multimédia.

La difficulté est ici de prendre en considération le grand nombre de paramètres associés aux différentes dimensions de la résolution des données et de les projeter sur des paramètres concernant les ressources, relatifs notamment au réseau (bande passante, latence, probabilité de déconnexion en micro-mobilité, …), à la base de données (relaxation partielle des propriétés ACID, degré de cohérence, …), et à la prise en compte de la sécurité (utilisation ou non d'un algorithme de cryptage, …). Afin de faciliter l'expression des demandes applicatives, nous nous proposons de regrouper ces paramètres en classes de service prédéfinies relatives chacune aux

différentes dimensions de la résolution. La définition de ces classes et leur validation est aujourd'hui un problème ouvert. Par ailleurs, l'architecture logicielle du code applicatif du côté du client et de la couche d'adaptation doit être définie plus précisément ; une architecture à base de composants/conteneurs, permettant de séparer clairement les aspects « métier » (application) des aspects non-fonctionnels (niveaux de résolution) peut être une approche intéressante.

10. Synthèse générale

L'informatique mobile promet la possibilité d'accéder à l'information et de la traiter n'importe où, n'importe quand et à partir de n'importe quel terminal. Cette omniprésence de l'information permet d'envisager de nouvelles applications, personnelles ou professionnelles, qui devraient avoir un impact sans précédent sur les systèmes d'information et leur utilisation. Par exemple, des employés itinérants pourraient accéder en permanence, où qu'ils se trouvent, des données relatives à leur entreprise ou à la tâche qu'ils exécutent en commun.

Les technologies de l'information et de la communication, et en particulier, les techniques de gestion de données réparties [Özsu et al, 1999], ont été conçues pour des clients et serveurs fixes, connectés par un réseau filaire. Les contraintes fortes des environnements mobiles (faible débit des réseaux hertziens, déconnexions fréquentes, faibles puissances des terminaux mobiles, etc.) changent radicalement les hypothèses sur lesquelles sont basées ces technologies. Ce constat a donné naissance à de nombreux problèmes de recherche, qui couvrent divers domaines comme les réseaux, les systèmes distribués, les bases de données, etc.

Dans cet article, nous avons fait un point sur l'état d'avancement des recherches en gestion de données dans les environnements mobiles. Afin de caractériser les différentes dimensions du problème, nous avons proposé une classification des applications mobiles, selon que la mobilité est liée à l'utilisateur, au serveur de données ou aux données, et introduit une architecture logicielle et matérielle de référence. En rapprochant les besoins des applications mobiles et les contraintes liées à l'architecture logicielle et matérielle, nous avons identifié les dimensions suivantes, pour lesquelles nous avons décrit la problématique, les solutions existantes et les perspectives de recherche:

- *Gestion de données spatio-temporelles :* la localisation des clients mobiles (téléphones cellulaires, flottes de véhicules …) et des données interrogées par les clients nécessite des bases de données qui prennent en compte le déplacement. Alors que le déplacement d'un objet mobile est généralement un événement continu, son reflet dans la base de données ne peut être que discret pour d'évidentes raisons de performance. Cette approximation a un impact fort sur la modélisation des données qui doit prendre en compte les dimensions spatiales et temporelles, sur les requêtes qui doivent prendre en compte les imprécisions et sur leur évaluation efficace qui nécessite des méthodes d'indexation ad-hoc. La gestion des données spatio-temporelles suit deux directions principales : les

bases de données contraintes avec un modèle de données abstrait non spécialisé et un langage de requêtes, et l'approche type abstrait de données avec des types spatiaux et des versions temporelles. Les problèmes ouverts concernent l'unification des approches de modélisation des objets mobiles, qui restent spécifiques à des domaines d'application, par ex. SIG, l'étude formelle des langages de requêtes et la validation expérimentale des techniques d'indexation.

- *Modèles d'accès à l'information :* le modèle client/serveur traditionnel est inadapté aux environnements mobiles, les serveurs constituant des goulots d'étranglement et des points de défaillance. D'autres modèles sont envisageables. Par exemple, le modèle *publish/subscribe* permet à un client d'être notifié automatiquement lorsqu'un événement satisfaisant un de ses abonnements est publié (émis). On peut également envisager qu'un serveur diffuse de façon répétitive une information et que les clients intéressés se mettent à son écoute. Ces modèles permettent d'optimiser la consommation de bande passante et de supporter l'asynchronisme lié aux déconnexions. Enfin, des modèles basés sur la diffusion épidémique de l'information permettent le passage à l'échelle (à un très grand nombre de clients) tout en augmentant la disponibilité des données (en évitant les points de défaillance). Les problèmes ouverts concernent l'amélioration des algorithmes de diffusion de l'information en augmentant la précision de la diffusion et ses performances.
- *Synchronisation et traitements déconnectés :* la mobilité peut être caractérisée par une alternance de phases de travail en mode connecté et en mode déconnecté. Des mises à jour peuvent ainsi intervenir sur des copies d'objets embarqués sur un calculateur mobile déconnecté. Il s'agit donc d'un modèle de réplication symétrique dans lequel les différentes copies ne sont pas accessibles de façon synchrone. La divergence de copie devient alors inévitable. Lors des reconnections, des protocoles de synchronisation doivent être appliqués pour rétablir la convergence des copies. Les principales solutions existantes sont des produits de synchronisation de fichiers ou de données, les gestionnaires de configuration et les outils de fusion avec des possibilités réduites de réconciliation. Une approche récente basée sur des algorithmes collaboratifs synchrones offre des possibilités de réconciliation sophistiquée. L'idée est d'exploiter la sémantique des opérations faites sur des objets typés en recourant à des transposées opérationnelles. Les problèmes ouverts concernent la définition d'un modèle de synchronisation générique, adaptable à tous types d'objets, le passage à l'échelle des algorithmes collaboratifs synchrones et l'approfondissement de la réconciliation dans le cadre des transposées opérationnelles.
- *Cohérence transactionnelle :* Les protocoles de synchronisation garantissent un degré de cohérence des données basé sur la propriété de convergence des copies. Les plus sophistiqués d'entre eux respectent également les règles de causalité (les opérations liées causalement sont exécutées dans le bon ordre) et de préservation de l'intention de l'utilisateur (sous réserve que celle-ci soit caractérisable). Cependant, certaines applications ont des besoins plus fort de

cohérence qui nécessitent le respect des propriétés transactionnelles ACID (Atomicité, Cohérence, Isolation, Durabilité). Les transactions sont initiées par des unités mobiles ou fixes, et leur exécution peut être distribuée sur l'ensemble des unités. La déconnexion d'une unité mobile et la durée non bornée de la déconnexion conduisent à reconsidérer les approches classiques au support des propriétés ACID. Parmi la diversité des solutions existantes, nous avons distingué les propositions où les unités mobiles initient des transactions qui s'exécutent sur le réseau fixe et les propositions où les transactions s'exécutent complètement ou partialement sur des unités mobiles. Les solutions proposées dans le cas de transactions s'exécutant sur une unité mobile restent limitées à des conditions d'applications spécifiques. Les problèmes ouverts concernent l'élaboration de solutions générales intégrant les différentes dimensions du problème et capables de s'insérer dans différents champs d'application.

- *Gestion des données embarquées :* Concevoir des composants bases de données destinés à être embarqués sur des calculateurs mobiles ultra-légers est un challenge important. En effet, chaque calculateur est architecturé de sorte à satisfaire des propriétés précises (portabilité, autonomie électrique, sécurité, adaptation à des traitements spécifiques…) en tenant compte de contraintes matérielles imposées (taille maximale de la puce, débit de communication contraint, technologie mémoire imposée…). Par ailleurs, ces architectures sont en permanente évolution pour couvrir les besoins de nouvelles applications. Les problèmes ouverts concernent la spécification de moteurs bases de données adaptés à ces architectures spécialisées, l'expression de recommandations pour les concepteurs de ces architectures dans une perspective de co-conception logicielle/matérielle, la définition de bancs d'essais adaptés à la gestion de données embarquées et la prise en compte de nouveaux types de données embarquées tels que XML.
- *Confidentialité des données :* lorsque les données sont trop volumineuses pour être embarquées ou qu'elles sont partagées entre plusieurs utilisateurs, le moyen le plus simple pour les rendre accessibles de tout endroit est de les stocker sur un serveur fixe, par exemple géré par un Database Service Provider et d'y accéder via l'Internet. Se pose alors le problème de la confidentialité de ces données. Des modèles de sécurité de bout en bout (du mobile à la donnée stockée) restent à définir afin de résister à tous types d'attaques, y compris celles dirigées vers l'empreinte disque de la base de données, qu'elles soient menées par un intrus ou par un administrateur de données. Les problèmes ouverts portent sur les techniques de chiffrement adaptées à un contexte base de données (partage, performance des traitements, résistance aux attaques statistiques), le recours à des environnements matériels sécurisés, la définition de modèles de droits d'accès dynamiques et enfin les principes de mise en œuvre de SGBD imposant le respect de clauses de confidentialité.
- *Adaptabilité :* l'objectif de l'adaptabilité est d'offrir à l'utilisateur, quelle que soit sa localisation et son type de terminal, un service aussi proche que possible de celui fourni dans un contexte statique, où le terminal de l'utilisateur est fixe,

dispose de ressources importantes et est relié aux serveurs par un réseau fiable et performant. L'adaptabilité a un impact à tous les niveaux de l'architecture logicielle: application, intergiciels, couche réseau et sources de données. Par exemple, la précision d'un objet téléchargé sur un terminal peut dépendre des capacités d'affichage et de stockage du terminal utilisé ainsi que du temps de connexion nécessaire au téléchargement. Le masquage de la déconnexion à l'utilisateur est également un souci majeur. Enfin, l'objectif étant de fournir à l'application le meilleur mode de fonctionnement en fonction de l'état courant des ressources, il faut être capable d'acquérir dynamiquement de l'information sur cet état. Des solutions répondent partiellement aux besoins d'adaptation dynamique en particulier : l'adaptation des données à la variabilité de la bande passante des réseaux sans fil, l'adaptation des applications au mode déconnecté, et l'adaptation des couches réseau à la micro-mobilité (permettant à un utilisateur de rester connecté à tout moment). Les problèmes ouverts concernent la représentation et l'expression de la résolution (qualité) des données, l'expression des demandes applicatives en termes de politiques, et les architectures d'adaptation dynamiques et leurs mécanismes.

Notre analyse des différentes dimensions du problème illustre la richesse et la pluridisciplinarité mais aussi les défis de la thématique « Mobilité et bases de données ». Chaque dimension offre ses propres perspectives de recherche. Au delà des améliorations des solutions existantes en termes de performances, passage à l'échelle, qualité, etc., nous pouvons retenir globalement la nécessité de généralisation des solutions, souvent limitées à un domaine d'application, d'étude formelle des modèles et langages proposés, et de validation et d'expérimentation des algorithmes en vraie grandeur. Enfin, l'introduction de nouvelles applications devrait apporter de nouveaux problèmes de recherche que nous ne pouvons identifier aujourd'hui.

Bibliographie

[Abdessalem et al, 2001] Abdessalem T., Moreira J., Ribeiro C., "Movement Query Operations for Spatio-Temporal Databases", Actes des Journées Bases de Données Avancées, Agadir, 2001.

[Agrawal et al, 2002] Agrawal R., Kiernan J., Srikant R., Xu Y., "Hippocratic Databases", Int. Conf. on Very Large Data Bases (VLDB), 2002.

[ALERT] Automated Local Evaluation in Real Time : http://www.alertsystems.org/

[Anciaux et al, 2001] Anciaux N., Bobineau C., Bouganim L., Pucheral P., Valduriez P., "PicoDBMS: Validation and Experience", Int. Conf. on Very Large Data Bases (VLDB), 2001.

[Bailey, 1975] Bailey N., «The Mathematical Theory of Infectious Diseases and its Applications», Second edition, Hafner Press, 1975.

[Balasubramaniam et al, 1998] Balasubramaniam S., Pierce B., "What is a File Synchroniser", Proc. 4th Annual ACM / IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'98), 1998.

[Baraani et al, 1996] Baraani A., Pieprzyk J., Safavi-Naini R., "Security In Databases: A Survey Study", TR-96-02, Department of Computer Science, The University of Wollongong, Wollongong, Australia, 1996. (ftp://ftp.cs.uow.edu.au/pub/papers/1995/tr-96-2.ps.Z)

[Birman et al, 1999] Birman K.P., Hayden M., Ozkasap O, Xiao Z, Budiu M, and Minsky Y. «Bimodal Multicast», ACM Transactions on Computer Systems, 17(2):41--88, May 1999.

[Bobineau et al, 2000] Bobineau C., Pucheral P., Abdallah M., "A Unilateral Commit Protocol for Mobile and Disconnected Computing". Int. Conf. on Parallel and Distributed Computing Systems (PDCS), 2000.

[Bonnet et al, 2000] Bonnet P., Gehrke J., Seshadri P., "Querying the Physical World", IEEE Personal Communications Special Issue on Networking the Physical World, 2000.

[Bonnet et al, 2001] Bonnet P., Gehrke J., Seshadri P., "Towards Sensor Database Systems". Mobile Data Management 2001: 3-14

[Bouganim et al, 2002] Bouganim L., Pucheral P., "Chip-Secured Data Access: Confidential Data on Untrusted Servers", Int. Conf. on Very Large Data Bases (VLDB), 2002.

[Cachin et al, 1999] Cachin C., Micali S., Stadler M., "Computationally Private Information Retrieval with Polylogarithmic Communication, Int. Conf. EUROCRYPT, 1999.

[Carrasco, 1999] Carrasco L. C.. RDBMS's for Java Cards ? What a Senseless Idea ! www.sqlmachine.com, 1999.

[Cherniack et al, 2001] Cherniack M., Franklin M.J., Zdonik S., « Expressing User Profiles for Data Recharging », IEEE Personal Communications, August 2001.

[Chrysanthis et al, 1993] Chrysanthis P. K., Transaction Processing in a Mobile Computing Environment, Proc. IEEE Workshop on Advances in Parallel and Distributed Systems, 1993,

[Conan et al, 2002] Conan, D., Chabridon, S., Bernard, G., « Disconnected Operations in Mobile Environments », 2nd Int. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, in Int. Parallel anf Distributed Processing Symposium (IPDPS 2002), 2002.

[Demers et al, 1987] Demers A, Greene D, Hauser C, Irish W, Larson J, Shenker S, Sturgis H, Swinehart D, and Terry D., «Epidemic Algorithms for Replicated Database Maintenance», ACM Symposium on Principles of Distributed Computing (PODC, 1987.

[Devanbu et al, 2000] Devanbu P., Gertz M., Martel C., Stubblebine S., "Authentic Third-Party Data Publication", Int. Conf. on Database Security (TBH), 2000.

[Dipert, 1997] Dipert B., FRAM: Ready to ditch niche ? EDN Access Magazine, Cahners Publishing Company, 1997.

[Domingo-Ferrer, 1997] Domingo-Ferrer J., "Multi-application smart cards and encrypted data processing", Future Generation Computer Systems, (13), 1997.

[Dourish, 1995] Dourish P., "The parting of the ways: Divergence, data management and collaborative work", European Conf. on Computer Supported Cooperative Work (CSCW), Mechanisms II, 1995.

[Dunham et al, 1997] Dunham M. H., Helal A., "A Mobile Transaction Model that Captures Both the Data and the Movement Behavior", ACM/Baltzer Journal on special topics in mobile networks and applications, Vol. 2, 1997.

[Dunham et al, 1998] Dunham M. H. and Kumar V., "Defining Location Data Dependency, Transaction Mobility and Commitment". Research Repport 98-CSE-01, Southern Methodist University, February 1998.

[Ellis et al, 1989] Ellis C.A., Gibbs S.J., "Concurency Control in Groupware Systems", ACM Int. Conf. on Management of Data (SIGMOD), 1989.

[Estublier, 1995] Estublier J., (editor) "Software Configuration Management", Selected Papers of the ICSE SCM-4 and SCM-5 Workshops, n° 1005 in Lecture Notes in Computer Science. Springer-Verlag, October 1995.

[Eugster et al, 2001] Eugster P., Guerraoui R., «Content-Based Publish/Subscribe with Structural Reflection», Usenix Conference on Object-Oriented Technologies and Systems (COOTS), 2001.

[Eugster et al, 2002] Eugster P, Guerraoui R., «Probabilistic Multicast», IEEE Int. Conf. on Dependable Systems and Networks (DSN), 2002.

[Feiler et al, 1990] Feiler P.F., Downey G.F., "Transaction-Oriented Configuration Management: A Case Study", Technical Report CMU/SEI-90-TR-23 ESD-90/TR-224, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, November 1990.

[Flinn et al, 1999] Flinn, J., Satyanaratanan, M., « Energy-aware adaptation for mobile applications », ACM Symposium on Operating Systems Principles, 1999.

[Giguère, 2001] Giguère E., “Mobile Data Management: Challenges of Wireless and Offline Data Access”, Int. Conf. on Data Engineering (ICDE), 2001.

[Gupta et al, 2001] Gupta I., Van Renesse R., and Birman K.P., «Scalable Fault-Tolerant Aggregation in Large Process Groups», IEEE Int. Conf. on Dependable Systems and Networks (DSN), 2001.

[Gray et al, 1996] Gray J., Helland P., O’Neil P., Shasha D., "Dangers of replication and a solution" , ACM Int. Conf. On Management of Data (SIGMOD), 1996.

[Grumbach et al, 1998] Grumbach S., Rigaux P., Segoufin L., "The DEDALE System for Complex Spatial Queries", ACM Int. Conf. On Management of Data (SIGMOD), 1998.

[Guting et al, 2000] Guting R., Bohlen P., Erwig M., Jensen C.S., Lorentzos N., Schneider M., Vazirgiannis M., "A Foundation for Representing and Querying Moving Objects", ACM Transactions on Database Systems (TODS), 2000.

[Hacigumus et al, 2002] Hacigumus H., Iyer B.R., Li C., Mehrotra S., “Executing SQL over Encrypted Data in the Database-Service-Provider Model”, ACM Int. Conf. On Management of Data (SIGMOD), 2002.

[He et al, 2001] He J., Wang M., “Cryptography and Relational Database Management Systems”, Int. Database and Engineering and Application Symposium, 2001.

[Herlihy et al, 1990] Herlihy M., Wing J., "Linearizability : a correctness condition for concurrent objects", ACM Trans. on Prog. Languages and Systems, Vol. 12, 3, 1990, p. 463-492.

[Holbrook et al 1995] Holbrook H.W., Singhal S.K., and Cheriton D.R., «Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation», ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication, (SIGCOMM), 1995.

[iButton, 2002] The crypto iButton with Java - (http://www.ibutton.com/)

[ISO, 1999] Int. Standardization Organization (ISO), Integrated Circuit(s) Cards with Contacts – Part 7: Interindustry Commands for Structured Card Query Language-SCQL, ISO/IEC 7816-7, 1999.

[Jing et al, 1999] Jing, J., Helal, A., Elmagarmid, A., « Client-server computing in mobile environments », ACM Computing Surveys, vol. 31, n° 2, 1999.

[Joseph et al, 1997] Joseph, A.D., Tauber, J.A., Kaashoek, M.F., « Mobile computing with the Rover toolkit », IEEE Transactions on Computers, vol. 46, n° 3, 1997.

[Karlsson et al, 2001] Karlsson J. S., Lal A., Leung C., Pham T., "IBM DB2 Everyplace: A Small Footprint Relational Database System", Int. Conf. on Data Engineering (ICDE), 2001

[Kim et al, 2001] Kim S., Hwang C.-S., Yu H., Lee S., Optimistic Scheduling Algorithm for Mobile Transactions Based on Reordering, Mobile Data Management, 2001.

[Kouznetsov et al, 2001] Kouznetsov P., Guerraoui R., Handurukande S.B., and Kermarrec A.-M., «Reducing Noise in Gossip-Based Reliable Broadcast», IEEE Symposium On Reliable Distributed Systems (SRDS), 2001.

[Kumar, 2000] Kumar V., "A Timeout-based Mobile Transaction Commitment Protocol", Proc. ADBIS-DASFAA, LNCS 1884, 2000.

[Lecomte et al, 1997] Lecomte S., Trane P., "Failure Recovery Using Action Log for Smartcards Transaction Based System", *IEEE On Line Testing Workshop*, 1997.

[Legrand et al, 2000], Le Grand G., Ben-Othman J., Horlait E., "Bandwidth Management in Mobile Environments with MIR (Mobile IP Reservation Protocol)", Int. Conf. On Network (ICON), 2000.

[Lin et al, 2000] Lin M. J., Marzullo K., and Masini S., «Gossip versus Deterministically Constrained Flooding on Small Networks». Int. Conf. on Distributed Computing Distributed Computing (DISC), 2000.

[Madden et al, 2002 (a)] Madden S., Franklin M., Hellerstein J., Hong W., "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks", Int. Conf on Operating Systems Design and Implementation, 2002.

[Madden et al, 2002 (b)] Madden S., Franklin M., Fjording the Stream: An Architecture for Queries over Streaming Sensor Data, Int. Conf. On Data Engineering (ICDE), 2002.

[Madria et al, 2001] Madria S. K., B. Bhargava, A Transaction Model for Improving Data Availability in Mobile Computing, Distributed and Parallel Databases, 2001.

[Marchiori, 2002] Marchiori M., editor, "The Platform for Privacy Preference 1.0 (P3P1.0) Specification. W3C proposed recommendation, 2002.

[Mattsson, 2000] Mattsson U., Secure.Data Functional Overview, Protegity Technical Paper TWP-0011, 2000. (http://www.protegrity.com/White_Papers.html)

[Molli et al, 2002] Molli P., Skaf-Molli H., Oster G., "Divergence Awareness for Virtual Team through the Web", Int. Conf. on Integrating Design and Process Technology (IDPT), 2002.

[Momin et al, 2000] Momin K.A., Vidyasankar K., "Flexible Integration of Optimistic and Pessimistic Concurrency Control in Mobile Environments", Lecture Notes in Computer Science 1884, Stuller et al. editor. 2000

[Munson et al, 1994] Munson J.P., Dewan P., "A flexible object merging framework", ACM Int. Conf. on Computer Supported Cooperative Work (CSCW), 1994.

[Noble et al, 1997] Noble, B.D., Satyanaratanan, M., Narayanan, D., Tilton, J.E., Flinn, J., Walker, K.R., « Agile Application-Aware for Mobility », ACM Symposium on Operating Systems Principles, 1997.

[Olsen, 1999] Olsen S., "Top Web Sites Compromise Consumer Privacy", CNET News Archive, 1999.

[Opyrchal et al, 2000] Opyrchal L., Astley M., Auerbach J., Banavar G., Strom R.E., and Sturman D.C., «Exploiting IP Multicast in Content-Based Publish-Subscribe Systems», Int. Conf. on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000), 2000.

[Oracle, 1999] Oracle Corp., Database Security in Oracle8i, 1999. otn.oracle.com/deploy/security/oracle8i

[Oracle, 2002] Oracle Corp, Oracle 9i Lite - Oracle Lite SQL Reference", Oracle Documentation, 2002.

[Ozkasap et al, 1999] Ozkasap O, van Renesse R, Birman K, and Xiao Z. « Efficient Buffering in Reliable Multicast Protocols », Int. Workshop on Networked Group Communication (NGC), 1999.

[Özsu et al, 1999] Özsu T., Valduriez P., Principles of Distributed Database Systems. Second edition, Prentice Hall, 1999.

[Perkins, 1996] Perkins, C.E., « Mobile-IP, Ad-Hoc Networking, and Nomadicity », Compsac'96, 1996.

[Pfoser et al, 1999] Pfoser D., Jensen C.S., "Capturing the Uncertainty of Moving-Object Representations", Int. Conf. on Scientific and Statistical Database Management, 1999.

[Pitoura et al, 1999] Pitoura E. and Bhargava B., "Data Consistency in Intermittently Connected Distributed Systems", Transactions on Knowledge and Data Engineering (TKDE), Vol. 11-6, 1999, p. 896-915.

[Pittel , 1987] Pittel B., "On Spreading of a Rumor". SIAM Journal of Applied Mathematics, 47:213--223, 1987.

[Pucheral et al, 2001] Pucheral P., Bouganim L., Valduriez P., Bobineau C., "PicoDBMS: Scaling down Database Techniques for the Smartcard", Very Large Data Bases Journal (VLDBJ), 10(2-3), 2001.

[Rivest et al, 1978] Rivest R. L., Adleman L., Dertouzos M. L., "On Data Banks and Privacy Homomorphisms", Foundations of Secure Computation. Academic Press, 1978.

[Saito et al, 2002] Saito Y., Shapiro M., "Replication : Optimistic Approaches", Technical Report HPL-2002-33, Hewlett-Packart Laboratories, 2002.

[Saltenis et al, 2002] Saltenis S., Jensen C.S., "Indexing of Moving Objects for Location-Based Services", Int. Conf. on Data Engineering (ICDE), 2002.

[Satyanarayanan, 1996] Satyanarayanan, M., « Mobile Information Access », IEEE Personal Communications, Février 1996.

[Schneier, 1996] Schneier B., Applied Cryptography, 2nd Edition, John Wiley & Sons, 1996.

[Serrano-Alvarado et al, 2001] Serrano-Alvarado P., Roncancio C. L., Adiba M. E.,"Analyzing Mobile Transactions Support for DBMS", Int. Workshop on Mobility in Databases and Distributed Systems (in DEXA), 2001.

[Seshadri et al, 2000] Seshadri P., Garrett P., "SQLServer for Windows CE - A Database Engine for Mobile and Embedded Platforms", Int. Conf. on Data Engineering (ICDE), 2000.

[SIRIUS] Etat du trafic Ile de France en temps réel : http://www.sytadin.equipement.gouv.fr/

[Smith et al, 1999] Smith S.W., Weingart S.H., Building a High-Performance, Programmable, Secure Coprocessor, Computer Networks (31) - 1999

[Suleiman et al, 1998] Suleiman M., Cart M., Ferrié J., "Concurrent Operations in a Distributed and Mobile Collaborative Environment", IEEE Int. Conf. on Data Engineering (ICDE), 1998.

[Sun et al, 1998] Sun C., Ellis C.S., "Operational Transformation in Real-Time Group Editors", ACM Int. Conf. on Computer Supported Cooperative Work (CSCW), 1998.

[Tichy, 1985] Tichy W.F., "RCS – A system for version control", Software–Practice and Experience, 15(7), July 1985, p. 637–654.

[Vidot et al, 2000] Vidot N., Cart M., Ferrié J., Suleiman M., "Copies convergence in a distributed real-time collaborative environment", ACM Int. Conf. on Computer Supported Cooperative Work (CSCW), 2000.

[Walborn et al, 1995] Walborn G.D., Chrysanthis P.K., " Supporting Semantics-Based Transaction Processing in Mobile Database Applications ", Symposium of Reliable Distributed Systems (SRDS), 1995.

[Walborn et al, 1999] Walborn G.D., Chrysanthis P.K., "Transaction Processing in PRO-MOTION", ACM Symposium on Applied Comupting, 1999.

[Wolfson et al, 1998] Wolfson O., Xu B, Chamberlain S., Jiang L., "Moving Objects Datababases: Issues and Solutions", Int. Conf. on Scientific and Statistical Database Management, 1998.

[XML, 2000] Xml diff and merge tool. Online http://alphaworks.ibm.com/, June 2000.

[Yeo et al, 1994] Yeo L.H., Zaslavsky A., "Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment", Int. Conf. on Distributed Computing Systems, 1994.